Abstractive Text Summarization: Enhancing Sequence-to-Sequence Models Using Word Sense Disambiguation and Semantic Content Generalization

Panagiotis Kouris
School of Electrical and Computer Engineering, National Technical University of Athens, Greece
pkouris@islab.ntua.gr

Georgios Alexandridis
School of Electrical and Computer Engineering, National Technical University of Athens, Greece
gealexandri@islab.ntua.gr

Andreas Stafylopatis
School of Electrical and Computer Engineering, National Technical University of Athens, Greece
andreas@cs.ntua.gr

Nowadays, most research conducted in the field of abstractive text summarization focuses on neural-based models alone, without considering their combination with knowledge-based approaches that could further enhance their efficiency. In this direction, this work presents a novel framework that combines sequence-to-sequence neural-based text summarization along with structure and semantic-based methodologies. The proposed framework is capable of dealing with the problem of out-of-vocabulary or rare words, improving the performance of the deep learning models. The overall methodology is based on a well-defined theoretical model of knowledge-based content generalization and deep learning predictions for generating abstractive summaries. The framework is composed of three key elements: (i) a pre-processing task, (ii) a machine learning methodology, and (iii) a post-processing task. The pre-processing task is a knowledge-based approach, based on ontological knowledge resources, word sense disambiguation, and named entity recognition, along with content generalization, that transforms ordinary text into a generalized form. A deep learning model of attentive encoder-decoder architecture, which is

Submission received: 30 April 2020; revised version received: 15 July 2021; accepted for publication: 28 July 2021.

https://doi.org/10.1162/COLI_a_00417

expanded to enable a coping and coverage mechanism, as well as reinforcement learning and transformer-based architectures, is trained on a generalized version of text-summary pairs, learning to predict summaries in a generalized form. The post-processing task utilizes knowledge resources, word embeddings, word sense disambiguation, and heuristic algorithms based on text similarity methods in order to transform the generalized version of a predicted summary to a final, human-readable form. An extensive experimental procedure on three popular data sets evaluates key aspects of the proposed framework, while the obtained results exhibit promising performance, validating the robustness of the proposed approach.

1. Introduction

The vast and constantly growing amount of textual information available online has rendered its access a challenging task and, as a consequence, it has increased the necessity for its ingestion in an automated manner. One of the principal ways of achieving this goal is through data reduction techniques that transform a piece of text into a succinct summary. **Text summarization** (TS), as this process is more formally known, has been an active research field for over half a century (Gambhir and Gupta 2017). The primary objective of automatic TS is to produce informative and human-readable summaries of documents, retaining their salient content. Since the advent of early work in the field of automatic TS (Luhn 1958; Edmundson 1969), several approaches and systems have been proposed, divided mainly into *single-document* TS (e.g., articles, news, stories, books, scientific papers, or weather forecast), *multi-document* TS (e.g., user reviews, news from several sources, or e-mails), and *query-based* TS (i.e., focusing on specific information from a text) (Nenkova and McKeown 2012).

Additionally, automatic TS techniques are further classified broadly into two groups; (i) *Extractive* TS and (ii) *Abstractive* TS (Yao, Wan, and Xiao 2017; Allahyari et al. 2017). The former aim at creating a summary by extracting a subset of sentences from the original text that include important informational aspects, minimizing redundancy. The latter aim at composing an abstract representation of the original text, using natural language generation to produce the summaries. In other words, an abstractive TS system generates new text, consisting of expressions, sentences, or words that might not have appeared originally, while at the same time incorporating the overall meaning of the initial document. Abstractive TS aims at generating high quality summaries in terms of cohesion, readability, and redundancy. Consequently, this is a challenging task, as it produces summaries that resemble or approximate human-written ones.

In general, abstractive TS approaches exhibit poor performance when compared with extractive TS (Gambhir and Gupta 2017; Joshi, Fernández, and Alegre 2018). Nevertheless and despite their weaknesses, abstractive TS systems are continually improving. Their main advantage is that of coping with the problems of cohesion, redundancy, and dangling anaphora, which are challenging to be tackled with extractive techniques. Furthermore, abstractive TS approaches produce concise summaries, reducing the size of the original sentences (i.e., applying sentence compression or sentence merging) and simultaneously generate coherent, grammatically correct, and readable summaries. One issue that affects abstractive TS are **out-of-vocabulary** (OOV) or rare words. This problem has a strong negative effect, especially on machine learning systems, which require training sets of sufficient usage examples for efficient predictions. Also, deep learning systems that achieve state-of-the-art performance in abstractive TS (Gupta and Gupta 2019) almost always fail to make accurate predictions when receiving new instances with rare or unseen words (i.e., words with few occurrences or words that are not included in the training set). In this sense, our work aims at providing a solution that deals with this category of words, aiding neural-based abstractive TS.

Particularly, this work focuses on abstractive TS of single documents, proposing a novel framework that utilizes knowledge-based **word sense disambiguation** (WSD) and semantic content generalization in order to enhance the performance of **sequence-to-sequence** (seq2seq) neural-based TS. The main contribution of this framework is the combination of the characteristics of three dominant aspects of abstractive TS and, more specifically, of structure, semantic, and neural-based approaches (Gupta and Gupta 2019) that are predominately treated as separate methodologies in the relevant literature (Section 2), especially in deep learning approaches. The presented framework, on the other hand, tries to unify them through the combined use of machine learning and knowledge-based techniques.

In this direction, the proposed methodology is composed of three distinct steps for producing the final summary; (i) the pre-processing task, (ii) the machine learning methodology, and (iii) the post-processing task. The first step achieves text generalization through the utilization of knowledge-based semantic ontologies and **named entity recognition** (NER) in order to extract named entities, concepts, and senses from the original document. Subsequently, the generalized text is provided to a seq2seq deep learning model of an attentive encoder-decoder architecture, which learns to predict a generalized version of the summary. In particular, five variants of the deep learning model are examined: (i) a seq2seq model with an attention mechanism, (ii) a pointer-generator network, (iii) a reinforcement learning model, (iv) a transformer approach, and (v) a pretrained encoder transformer architecture (Section 5). Lastly, the post-processing task creates the final summary, based on heuristic algorithms and text similarity metrics that match the concepts of the generalized summary to specific ones. The extensive experimental procedure conducted on three widely used data sets (`Gigaword` [Napoles, Gormley, and Van Durme 2012], `Duc 2004` [Over, Dang, and Harman 2007], and `CNN/DailyMail` [Hermann et al. 2015]) yields promising results, alleviating the problem of rare and OOV words and outperforming state-of-the-art seq2seq deep learning techniques.

The rest of this paper is organized as follows: Section 2 overviews the relevant literature. Section 3 outlines the proposed framework, which is further analyzed in Section 4 (the pre-processing task), Section 5 (the machine-learning methodology), and Section 6 (the post-processing task). Section 7 describes the experimental procedure, and Section 8 presents the obtained results, which are discussed in Section 9. Finally, Section 10 concludes this work, with some final remarks and future work directions.

2. Related Work

Early approaches to abstractive TS included sentence compression (Knight and Marcu 2002; Cohn and Lapata 2008) and fusion (Barzilay and McKeown 2005; Marsi and Krahmer 2005; Filippova and Strube 2008; Filippova 2010), which combine similar phrases of the input sentences, generating a concise version of them. Most current approaches, however, are either structure- or semantic-based (Moratanch and Chitrakala 2016), and in recent years, a third category has emerged; that of neural-based abstractive TS (Gupta and Gupta 2019). In particular, structure-based methods exploit predefined structures, like trees, ontologies, graphs, rules, and templates in order to create an abstractive summary. Semantic-based approaches, on the other hand, use natural language generation systems, based on semantic-graphs, predicate-arguments, and information items for summary generation, utilizing the semantic representation of input text. Finally, neural-based approaches utilize deep learning networks to predict abstractive summaries and have become dominant in abstractive TS, as they achieve state-of-the-art performance (Lin and Ng 2019; Gupta and Gupta 2019).

Structure-based methods commonly rely on hierarchical ontologies as knowledge sources, which can be used for the semantic representation of a document, resolving ambiguity issues (Mohan et al. 2016). In ontology-based approaches, knowledge resources such as *WordNet* (Miller 1995; Fellbaum 1998), *DBPedia* (Bizer et al. 2009), or other domain-specific ontologies are used to create abstractive summaries (Mohan et al. 2016). Several ontology-based abstractive systems have been developed that extract concepts or key phrases from a text, creating a summary (Lee, Jian, and Huang 2005; Hennig, Umbrath, and Wetzker 2008; Baralis et al. 2013; Hípola et al. 2014). In this work, a pre-defined ontology of concepts is utilized for content generalization in the pre-processing phase and for concept matching in the post-processing phase, as it is going to be described in the following sections.

Semantic-based approaches utilize a (semantic) representation of text, which captures the relations between entities in order to identify significant sentences, phrases, or expressions that are accordingly combined to create the summary. In particular, semantic graph-based methods convert the text to a graph representation, which captures semantic and syntactical relations (Khan et al. 2018; Joshi, Wang, and McClean 2018). Usually, the summary is generated by finding significant concepts, either utilizing graph relationships or reducing the size of the graph by eliminating redundancy or rejecting non-significant entities (Moawad and Aref 2012). Furthermore, item-based methods use information items (e.g., triplets of subject, verb, and object) to generate sentences. The summary is shaped by ranking the sentences according to the information items they contain (Genest and Lapalme 2011). Moreover, predicate-argument-based solutions obtain similar structures of predicate-arguments (i.e., subjects, verbs, and objects), merging them semantically to create the summary (Alshaina, John, and Nath 2017; Zhang, Zhou, and Zong 2016). Because the approaches of this category cannot reach the level of performance of deep learning techniques in abstractive TS (Gupta and Gupta 2019), the present work combines semantic and neural-based methodologies in order to improve the efficiency of the former. On a semantic basis, the proposed framework utilizes an ontology that transforms text into a generalized version by identifying concepts and their relationships.

In contrast to the methods described above, which require challenging subtasks such as information extraction, content selection, and natural language generation (Lin and Ng 2019), neural-based approaches are capable of producing abstractive summaries, using only an appropriate neural network model, without other complicated natural language processing. Such techniques are often based on seq2seq models of encoder-decoder architecture (Sutskever, Vinyals, and Le 2014); they are deep learning models, where the encoder takes an input sequence of words and the decoder emits an output sequence of words that constitutes the summary. The said networks are trained end-to-end on a large corpus of text-summary pairs, learning to predict abstractive summaries of input text. The deep learning models are mainly based on **recurrent neural networks** (RNN), and, primarily, on **long short-term memory** (LSTM), **gated recurrent units** (GRU), and transformer-based architectures that accomplish state-of-the-art performance in abstractive TS (Chopra, Auli, and Rush 2016; Nallapati et al. 2016; See, Liu, and Manning 2017; Song, Huang, and Ruan 2018; Gao et al. 2020; Lin and Ng 2019; Vaswani et al. 2017; Liu and Lapata 2019; You et al. 2019; Xu et al. 2020; Wolf et al. 2020).

The basic seq2seq model may be further improved through the introduction of an *attention mechanism* (Bahdanau, Cho, and Bengio 2014), thereby making the attentive encoder-decoder models a standard architecture for seq2seq neural networks (Luong, Pham, and Manning 2015) and especially for neural-based abstractive TS (Nallapati et al. 2016; See, Liu, and Manning 2017; Cohan et al. 2018; Lin and Ng 2019). The attention mechanism focuses on important words; given the encoder input sequence, the decoder is fed with a context vector that quantifies their importance. Moreover, encoder-decoder models with attention mechanisms have been used to face the problem of OOV words through the addition of a pointer generator network (Nallapati et al. 2016; See, Liu, and Manning 2017). Also, See, Liu, and Manning (2017) support attention with a coverage mechanism, avoiding repetition of the same words in the summary. Additionally, Lin et al. (2018) present a global encoding model for solving the problem of repetition. Furthermore, Song, Huang, and Ruan (2018) propose a model of extracting phrases from text to generate summaries, which is based on deep LSTM units and **convolutional neural networks** (CNN). An abstractive TS framework, based on an transformer-based encoder-decoder architecture with a focus-attention mechanism and a model for selecting important aspects to distinguish salient information creating summaries, is presented in You et al. (2019). Finally, the transformer-based models and their pretrained variants constitute the dominant approaches in TS, which are based in simple models that incorporate self-attention in order to reduce the computational load, parallelizing the computations in the training process (Vaswani et al. 2017; Zhang, Xu, and Wang 2019; Liu and Lapata 2019; You et al. 2019; Xu et al. 2020; Pilault et al. 2020; Wolf et al. 2020).

The main weakness of the above-described encoder-decoder architectures is that they minimize a maximum-likelihood loss function, while TS evaluation is based on a different metric (i.e., ROUGE scores). To address this problem, recent approaches utilize **reinforcement learning** (RL) (Li 2018; Keneshloo et al. 2020) to maximize a reward based on the employed evaluation metric. In particular, Paulus, Xiong, and Socher (2018) propose a RL seq2seq model that optimizes a specific evaluation metric (e.g., ROUGE-L score) and a mixed training objective model, which combines the optimization of the maximum-likelihood loss and the evaluation metric in question. Celikyilmaz et al. (2018) present a RL model for long documents that uses multiple collaborating encoder agents for encoding different sections (e.g., sentences or paragraphs) of the text, where the communication between agents improves summarization. Pasunuru and Bansal (2018) introduce a multiaward approach to improve the saliency and directed logical entailment in abstractive TS. In the current work, the approach of Paulus, Xiong, and Socher (2018) is adopted in introducing a RL model to our framework.

As can be seen above, the majority of neural-based approaches try to improve the deep learning models without taking advantage of semantic or structure-based techniques. Instead, this work, significantly extending an earlier approach (Kouris, Alexandridis, and Stafylopatis 2019), proposes a novel knowledge-based framework that enhances neural-based abstractive TS, by combining characteristics of structure and semantic-based methodologies. In particular, in the current work, the content generalization methodology has been redesigned through the inclusion of WSD for more accurate concept generalization, assuming new generalization strategies. Moreover, the proposed deep learning model has been extended to include a coping and coverage mechanism, as well as reinforcement learning to improve performance and, simultaneously, to examine the versatility of our framework in different seq2seq models. Additionally, the post-processing task, which is equipped with an improved greedy and a new optimal solution, is based on WSD and the employed ontology to perform concept matching for generating the final summary. The basic unit of the said framework is an appropriate sequence-to-sequence deep learning neural network, aided by a knowledge-based methodology, as is described in detail below.

3. Framework Overview

The overall framework is illustrated in Figure 1. The input comprises a single-document *text*, along with a taxonomy of concepts *T*, while the output is a human-readable *summary*. Its main components are five, starting with WSD, whose purpose is to generalize ambiguous words. This is an important step for increasing the accuracy of content generalization that follows next and deals with OOV or rare words. Both of the aforementioned steps constitute the pre-processing phase, which is discussed in more detail in Section 4.

The generalized text is subsequently mapped to a continuous vector space, using *neural language processing* techniques. The said vectors are then provided to a deep seq2seq model of encoder-decoder architecture, which additionally incorporates an attention mechanism. The model, having been trained on a corpus of text-summary pairs, predicts a generalized summary for the new input it has been given. Both the vector-mapping and deep learning prediction steps are components of the machine learning phase and are further analyzed in Section 5.

Lastly, the produced, generalized summary of the previous steps is post-processed in order to obtain its final text in human-readable form. This involves a number of individual steps like identifying potentially new generalized concepts introduced in the resulting summary and how these are optimally matched to concepts in the original document. The intuition behind this final phase is further reasoned upon in Section 6.

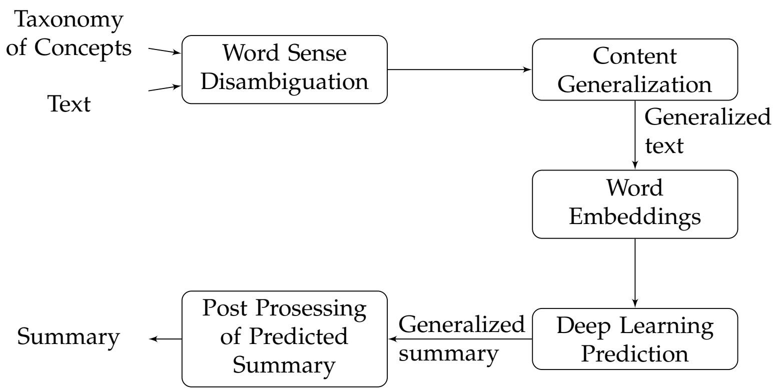


Figure 1
The workflow of the proposed framework.

4. Pre-processing Phase

As outlined above, the pre-processing phase consists of two components: WSD, discussed in Section 4.1 below, and content generalization, presented in Section 4.2. In particular, content generalization strategies are examined in more detail in Section 4.3.

4.1 Word Sense Disambiguation

WSD is the process of determining the exact sense of a word in a particular context, thereby assessing the different senses the same words might have (Navigli 2009; Borah, Talukdar, and Baruah 2014). The purpose of including this task in the presented framework is twofold; first, to generalize a particular sense of a word to a more general concept (Section 4.2) and second, to exploit this technique in order to transform a text into its disambiguated version. Additionally, WSD is expected to improve the performance of the seq2seq model (Section 5), limiting the search space of candidate sequences of words for the final summary. More specifically, a set of disambiguated text-summary pairs are used in the training phase of the seq2seq model and the latter learns to predict disambiguated summaries of input text.

Sense representation is accomplished in *WordNet* (Miller 1995; Fellbaum 1998) through the use of a WSD identifier like, for example, in *book.n.01*, *book.n.02*, *watch.v.02*, or *watch.v.06*. The same notation is also adopted in this work, consisting of three parts separated by dots; (i) a word, (ii) a part of speech (i.e., *n* for nouns and *v* for verbs), and (iii) a sense number (e.g., *01, 02, 06*). For instance, the word "*bank*" has more than one sense, such as *bank.n.01*, *bank.n.02*, or *bank.v.01*, where the first two are nouns with different meanings (since their sense numbers are different), while the third one represents a verb. To disambiguate a document, its words are transformed into their respective WSD identifiers (e.g., *table* into *table.n.01*), according to their sense.

Because the overall framework is a knowledge-based system, a WSD technique based on knowledge resources for deciding the particular senses of words (Navigli 2012; Chaplot and Salakhutdinov 2018) is the most natural choice. Nevertheless, other approaches might also be considered, such as supervised WSD (Raganato, Camacho-Collados, and Navigli 2017), provided that there exist sufficient WSD learning corpora. In the experimental part of this work (Section 7), the effect of WSD in the utilized seq2seq model is further investigated.

4.2 The Theoretical Model of Content Generalization

Content (or text) generalization constitutes the main effort this framework puts in improving the performance of machine learning predictions. Its main purpose is to deal with OOV or rare words; in particular, a word not existing in the trained system's vocabulary can be generalized to a more general one, with a sufficient presence in the training set. In a similar manner, rare words, with a small frequency in the training set, can be generalized to more frequent ones.

Prior to discussing the specifics of the achieved content generalization, a theoretical model must be defined and have its properties studied and reasoned upon. In principle, the process of content generalization requires a taxonomy of concepts (like, for example, those appearing in Figure 2), as described in Definition 1. A taxonomy of concepts is based on a knowledge resource (e.g., WordNet) and consists of words or concepts and their semantic relationship in a hierarchical structure. More specifically, the studied

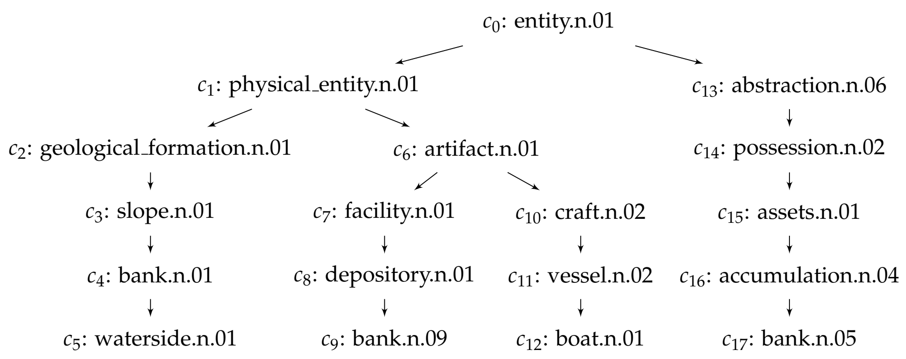


Figure 2
An example of a taxonomy of concepts.

taxonomy is comprised of disambiguated senses that represent a unique meaning of a word that is alternatively called a concept.

Definition 1 (Taxonomy of Concepts)
A taxonomy of concepts consists of a hierarchical structure of semantically disambiguated concepts that are related to their subordinate or superordinate concepts with an `is-a` type of relationship.

The relations among concepts are described with the use of the terms **hyponym** (Definition 2) and **hypernym** (Definition 3). A hyponym, or a subordinate of a word, entails a more specific semantic field of the given word (e.g., the horse is a hyponym of animals). On the other hand, a hypernym or a superordinate of a word, has a broader semantic field than the given word (e.g., the vehicle is a hypernym of bus). The **root** of the taxonomy corresponds to the word of the broadest or the most general semantic field (e.g., entity), while the **leaves** contain the most specific concepts.

Definition 2 (Hyponym)
Given a taxonomy of concepts, a concept c_b is a hyponym of a concept c_a iff c_b semantically entails c_a ($c_b \models c_a$).

Definition 3 (Hypernym)
Given a taxonomy of concepts, a concept c_a is a hypernym of a concept c_b iff c_b semantically entails c_a ($c_b \models c_a$).

Furthermore, the hyponym and hypernym path of concepts are introduced in Definitions 4 and 5, respectively. Because the taxonomy of concepts is assumed to have a hierarchical structure (i.e., a tree form), a hyponym path of concepts represents the taxonomy path from a particular concept to the leaf of the taxonomy, whereas a hypernym path of concepts includes the concepts from a specific concept to the root of the taxonomy.

Definition 4 (Taxonomy Path of Hyponyms)
Given a taxonomy of concepts, a taxonomy path of hyponyms of a concept c_a is an ordered sequence of concepts $\{c_a, c_{a+1}, ...c_i, c_{i+1}, ...c_{n-1}, c_n\}$, where concept c_{i+1} semantically entails concept c_i and c_n is a leaf concept of the taxonomy.

Definition 5 (Taxonomy Path of Hypernyms)
Given a taxonomy of concepts, a taxonomy path of hypernyms of a concept c_a is an ordered sequence of concepts $\{c_a, c_{a-1}, ... c_{i+1}, c_i, c_{i-1}, ... c_{r+1}, c_r\}$, where c_{i+1} semantically entails c_i and c_r is the root concept of the taxonomy.

Once the concepts have been extracted from text, their hypernym taxonomy path is used for generalization, while the taxonomy paths of hyponyms are used to make the extracted terms more specific. Both a path of hypernyms and of hyponyms may be used to move from a specific concept to a general one and from a general concept to a specific one, respectively (i.e., generalizing or specifying text).

Example 1 (Taxonomy of Concepts)
Figure 2 illustrates a sample taxonomy of 18 concepts, capturing the relations between them. Each node contains a WSD identifier (e.g., *facility.n.01*) and the word *"bank"* appears in three different senses; *bank.n.01* (sloping land), *bank.n.05* (a supply or stock for future use), and *bank.n.09* (a building in which the respective business is located). The taxonomy path of the hyponyms of concept c_3 (*slope.n.01*) is $\{c_3, c_4, c_5\}$, while the taxonomy path of the hypernyms of the same concept is $\{c_3, c_2, c_1, c_0\}$. The most general concept, with the broadest semantic field, of this taxonomy is that of *entity.n.01*, which is called the root concept. This taxonomy contains nouns exclusively; a different one, for example, might have contained verbs.

Definitions 6 and 7 below define when a concept is assumed to be generalizable and when a piece of text is generalizable, respectively. A concept c_i with a taxonomy path of hypernyms Ph_{c_i} is generalizable when c_i semantically entails a concept $c_j \in Ph_{c_i}$ with a smaller taxonomy depth than c_i (i.e., c_i can be exchanged with c_j). Also, a text is generalizable when one or more of its concepts are generalizable.

Definition 6 (Generalizable Concept)
A concept c_i is generalizable when its taxonomy path of hypernyms Ph_{c_i} contains at least one concept $c_j \in Ph_{c_i}$ such that the concept c_i semantically entails c_j.

Definition 7 (Generalizable Text)
A piece of text is generalizable when at least one concept it contains is generalizable.

A taxonomy may contain concepts of a very general meaning, like *entity* or *object*. To avoid generalizing to such general concepts, a level of generalization needs to be defined, which sets limits on the degree of generalization. Definition 8 specifies that a concept may be generalized to a minimum taxonomy depth, according to the given level of generalization.

Definition 8 (Level of Generalization)
The level of generalization of a text is equal to the minimum taxonomy depth that a concept can be generalized.

4.3 Generalization Strategies

Based on the theoretical model presented above, six distinct generalization strategies are being studied, as a pre-processing step for the machine learning model (Section 5). In particular, the methods outlined herein investigate how each strategy enhances machine learning-based TS and whether WSD provides additional improvement or not. The examined strategies follow two different directions: (i) generalizing concepts to their hypernyms, using a semantic ontology of concepts, and (ii) generalizing concepts to their named entities, using NER. Additionally, the combination of the aforementioned approaches is also considered and the generalization strategies are applied to disambiguated text using WSD.

The examined techniques also take into account the frequency of each term (θ_f) in the source text. The intuition behind this choice is the fact that supervised machine learning systems require sufficient numbers of usage examples for efficient training. In this sense, generalizing OOV or rare words to more frequent ones may result in more accurate summaries, thereby improving the predictions of the machine learning system. Of course, the determination of θ_f is a hyperparameter of the overall approach and in the experiments (Section 7) its optimum value is ascertained with respect to the utilized data sets.

A second hyperparameter that affects the operation of the examined strategies is the level of generalization θ_d (Definition 8), which prohibits concepts from being generalized to very broad meanings. Over-generalization is not expected to have any positive effect on supervised machine learning, as the system will not be able to discriminate between the different meanings of the same very general concepts, failing to predict an appropriate summary.

4.3.1 Level-Driven Generalization (LG). This strategy generalizes concepts according to (i) the threshold θ_f, under which concepts should be generalized and (ii) the given level of generalization θ_d. Example 2 describes the LG strategy, where the concepts of a sentence are generalized to their hypernyms according to their frequency in the training set.

Example 2 (LG)
Given

1. the sentence *"he is sitting on the bank of the river watching a boat"*
2. its disambiguated version: *"he is sitting on the bank.n.01 of the river.n.01 watching a boat.n.01"*
3. the taxonomy of concepts T of Figure 2

4. a set of concepts with their frequency in the overall training set (not just the sentence)

$$F = \{(\text{“}bank.n.01\text{”},\ 58), (\text{“}slope.n.01\text{”},\ 120),\ (\text{“}boat.n.01\text{”},\ 45),$$
$$(\text{“}vessel.n.02\text{”},\ 98),\ (\text{“}craft.n.02\text{”},\ 160),\ (\text{“}river.n.01\text{”},\ 220)\}$$

5. thresholds $\theta_d = 3, \theta_f = 100$

Then the candidate concepts for generalization are *bank.n.01* and *boat.n.01*, as their frequency in the data set is less than θ_f and their taxonomy depth in T is greater than θ_d.

The hypernym paths of these concepts are extracted from taxonomy T as follows:

$$P_{bank.n.01} = \{\text{“}bank.n.01\text{”},\ \text{“}slope.n.01\text{”},\ \text{“}geological\_formation.n.01\text{”},$$
$$\text{“}physical\_entity.n.01\text{”},\ \text{“}entity.n.01\text{”}\}$$
$$P_{boat.n.01} = \{\text{“}boat.n.01\text{”},\ \text{“}vessel.n.02\text{”},\ \text{“}craft.n.02\text{”},\ \text{“}artifact.n.01\text{”},$$
$$\text{“}physical\_entity.n.01\text{”},\ \text{“}entity.n.01\text{”}\}$$

Thus, the sentence is generalized to: *"he is sitting on the slope.n.01 of the river watching a craft.n.01"*, satisfying the given thresholds. It should also be noted that the generalized concepts are represented by WSD identifiers for easier recognition in the post-processing phase, as it shall be described next.

Algorithm 1 outlines the steps of the LG content generalization strategy. It requires an input *text*, its disambiguated version *wsdText*, a set of concept frequencies F, and the thresholds θ_d, θ_f discussed above. In the for-loop (lines 2–20), all tokens of input text are accessed, with those having frequency less than θ_f becoming candidate for generalization (line 4). In this case, the disambiguated version c of *token* is retrieved, along with its path of hypernyms P_c, its taxonomy depth d_c, and its frequency c (lines 5–8). As long as the frequency and the taxonomy depth of the currently selected concept c do not meet the threshold requirements, it is updated by the next hypernym in the path (lines 9–13). If the previous loop has resulted in the retrieval of a generalized concept c (line 14), then *token* is replaced by c (line 15) and the frequencies of *token* and c are updated (lines 16–17). Once this procedure is executed for all tokens in *text*, the algorithm terminates, returning the generalized version of *text* (*genText*, line 21).

The computational complexity of Algorithm 1 is $O(k \cdot n)$ because, in the worst case, all n input tokens will be examined (i.e., $f_{token} \leq \theta_f\ \forall\ token \in text$). For each input token, the while loop of lines 9–13 examines all concepts of the longest taxonomy path ($k = |P_{token}|$). In practice, $k << n$ (especially for long texts), therefore the computational complexity of Algorithm 1 approaches $O(n)$; a linear complexity that allows the algorithm to perform at low running times.

4.3.2 WSD-Based Level-Driven Generalization (W-LG). This strategy is a modification of LG, as the input text is first disambiguated prior to being provided as input to

Algorithm 1 Level-driven text generalization (LG)

```
Require: $text$, $wsdText$, $F$, $T$, $\theta_d$, $\theta_f$
 1: $genText \leftarrow text$
 2: for all $token \in text$ do
 3:     $f_{token} \leftarrow$ Frequency of $token$ from $F$
 4:     if $f_{token} \leq \theta_f$ then
 5:         $c \leftarrow$ WSD of $token$ form $wsdText$
 6:         $P_c \leftarrow$ Path of hypernyms of $c$ from $T$
 7:         $d_c \leftarrow$ taxonomy depth of concept $c$
 8:         $f_c \leftarrow$ Frequency of $c$ from $F$
 9:         while $f_c < \theta_f$ and $d_c > \theta_d$ do
10:             $c \leftarrow$ hypernym of $c$ from $P_{token}$
11:             $d_c \leftarrow$ taxonomy depth of $c$
12:             $f_c \leftarrow$ Frequency of $c$ from $F$
13:         end while
14:         if (word of $c$) $\neq token$ then
15:             $genText \leftarrow$ generalize $token$ of $genText$ to $c$
16:             $F \leftarrow (F \setminus \{(token, f_{token})\}) \cup \{(token, f_{token} - 1)\}$
17:             $F \leftarrow (F \setminus \{(c, f_c)\}) \cup \{(c, f_c + 1)\}$
18:         end if
19:     end if
20: end for
21: return $genText$
```

Algorithm 1. Similar to LG, infrequent concepts (not satisfying threshold θ_f) are being generalized. W-LG is further illustrated in Example 3, which is based on Example 2.

Example 3 (W-LG)
Based on the input text and the other parameters of Example 2, the disambiguated sentence "*he is sitting on the bank.n.01 of the river.n.01 watching a boat.n.01*" is provided to Algorithm 1 and is subsequently generalized to "*he is sitting on the slope.n.01 of the river.n.01 watching a craft.n.01*".

The output also retains the WSD identifiers of those concepts that are not generalized, like *river.n.01*.

4.3.3 Named Entity-Driven Generalization (NEG). This strategy generalizes concepts according to (i) particular named entities (e.g., location, person, organization) that they represent and (ii) a minimum frequency θ_f, under which they become candidate for generalization. NEG is presented in more detail in Example 4, where the concepts in a sentence are generalized to their named entities, according to their frequency in the training set.

Example 4 (NEG)
Given:

1. the sentence "*Elizabeth works at an antique shop in New York City*"
2. a set of named entities $E = \{LOCATION, PERSON, ORGANIZATION\}$

3. a set of concepts with frequencies in the training set
   $F = \{(\text{"}\textit{Elizabeth}\text{"}, 58), (\text{"}\textit{antique shop}\text{"}, 22), (\text{"}\textit{New York City}\text{"}, 140)\}$
4. $\theta_f = 100$

the recognized named entities are the following: (i) *"Elizabeth" (PERSON)*, (ii) *"antique shop" (ORGANIZATION)* and (iii) *"New York City" (LOCATION)*. The candidate concepts for generalization are *"Elizabeth"* and *"antique shop"*, as their frequency in the training set is below the given threshold θ_f. Thus, the sentence is generalized to *"PERSON works at an ORGANIZATION in New York City"*.

Algorithm 2 Named entities-driven text generalization (NEG)

```
Require: $text$, $E$, $F$, $\theta_f$
1: $genText \leftarrow text$
2: $tokenNamedEntities \leftarrow$ named entinties of $text$ from $E$
3: for all ($token$, $namedEntity$) $\in$ $tokenNamedEntities$ do
4:     $f_{token} \leftarrow$ Frequency of $token$ from $F$
5:     if $f_{token} < \theta_f$ then
6:         $genText \leftarrow$ generalize $token$ of $genText$ to $namedEntity$
7:     end if
8: end for
9: return $genText$
```

Algorithm 2 describes the steps of the NEG strategy in more detail. The inputs are the candidate *text* for generalization, a set of named entities E, a set of concepts F, and the frequency threshold θ_f. Initially, the generalized text *genText* is set to be identical to the original (line 1) and, subsequently, the task of NER is applied to *text*, which results in the determination of tuples of type (token, named entity) in the input text (*tokenNamedEntity*, line 2). Then, all of the aforementioned tuples in *tokenNamedEntity* are accessed (lines 3–8) and the frequency f_{token} of each token is retrieved (line 4). If the said frequency is below threshold θ_f (line 5), then the examined *token* is generalized to its respective name-entity in the (generalized version) *genText* of text (line 6). The algorithm terminates when all *tokenNamedEntity* pairs have been examined, returning the generalized version *genText* of input text. The computational complexity of Algorithm 2 is linear ($O(n)$) because, in the worst case, the NEG generalization strategy examines all n input tuples ($token, namedEntity$).

4.3.4 WSD-Based Named Entity-Driven Generalization (W-NEG). This strategy is a modification of NEG, in the same way that W-LG is a modification of LG; instead of providing the original text as input to Algorithm 2, it is exchanged by its disambiguated version. This is better illustrated in Example 5, which is an adaptation of Example 4 for this strategy.

Example 5 (W-NEG)
Based on the input text and the other parameters of Example 4, the disambiguated sentence *"Elizabeth works at an antique.n.02 shop.n.01 in New_York_City.n.01"* is generalized to *"PERSON works at an ORGANIZATION in New_York_City.n.01"*.

The named entities correspond to words or phrases identified by NER in the original text that have replaced the original concepts in the text generalization phase.

Also, the generalized text may retain the WSD identifiers of those concepts that are not generalized, like *New_York_City.n.01* in the above sentence.

4.3.5 Combination of NEG and LG (NEG-LG). This strategy performs content generalization in two steps, first applying NEG and then LG. Example 6 showcases the application of the NEG-LG strategy.

Example 6 (NEG-LG)
Let's assume we are given the sentence *"Elizabeth, who works at an antique shop in New York City, is sitting on the bank of the river watching a boat"*, along with the other parameters of Examples 4 and 2, with the exception of adding in F the term *"who"*, which has 520 occurrences in the training set.

The named entities of this sentence, as determined by NER, are the following: (i) *"Elizabeth" (PERSON)*, (ii) *"who" (PERSON)*, (iii) *"antique shop" (ORGANIZATION)*, and (iv) *"New York City" (LOCATION)*.

Because the NEG strategy is applied first, the candidate concepts for generalization are *"Elizabeth"* and *"antique shop"*, as their frequency in the training set is below the given threshold θ_f. Therefore, the disambiguated version or the sentence becomes: *"PERSON, who works at an ORGANIZATION in New York City, is sitting on the bank.n.01 of the river.n.01 watching the boat.n.01"*.

Next, the LG strategy generalizes concepts *bank.n.01* and *boat.n.01* because their individual frequencies in the training set are below threshold θ_f. As a result and after the application of both techniques, the sentence is generalized to *"PERSON, who works at an ORGANIZATION in New York City, is sitting on the slope.n.01 of the river watching the craft.n.01"*.

4.3.6 WSD-Based Combination of NEG and LG (W-NEG-LG). The final generalization strategy to be examined combines NEG and W-LG. It is described in Example 7.

Example 7
[W-NEG-LG] Given the sentence *"Elizabeth, who works at an antique shop in New York City, is sitting on the bank of the river watching a boat"* and its disambiguated version *"Elizabeth, who.n.01 works at an antique.n.02 shop.n.01 in New_York_city.n.01, is sitting on the bank.n.01 of the river.n.01 watching a boat.n.01"* and the parameters of Example 6, then it is generalized to *"PERSON, who.n.01 works at an ORGANIZATION in New_York_city.n.01, is sitting on the slope.n.01 of the river.n.01 watching the craft.n.01"*, according to W-NEG-LG.

The generalized sentence retains the WSD identifiers of those words that are not generalized.

5. Machine Learning

The machine learning phase of the proposed framework comprises two distinct steps. First, the generalized text, obtained from the previous phase, is mapped to a continuous vector space, using NLP techniques (Section 5.1). Then, the retrieved vectors are provided as input to the deep seq2seq model that learns to predict the summary (Section 5.2).

5.1 Word Representation

In principle, word representation techniques based on *neural language processing* map each token to a vector of real values of dimensionality D, formally known as a **word embedding** (Li and Yang 2018; Mikolov et al. 2013; Pennington, Socher, and Manning 2014; Bojanowski et al. 2017). The main difference of word embeddings from other vector-based representations of text is that the former retain the semantic relationship between words, in the sense that words with similar meanings are mapped closer in the embedding space.

The proposed framework makes use of word embeddings in model training (Section 5.2), where the text-summary pairs of a particular generalization strategy are mapped to their respective vectors, and in the post-processing phase (Section 6). The framework itself is *neural language processing*-agnostic, in the sense that any word embedding methodology may be used like word2vec, GloVe, or fastText (Pennington, Socher, and Manning 2014; Mikolov et al. 2017), as long as it retains the semantic relationships of words in the vector space. In the experimental part of this work (Section 7), word2vec has been used and, more specifically, the **continuous bag-of-word** model (Mikolov et al. 2013; Rong 2014).

Even though the proposed framework is agnostic with respect to the chosen *neural language processing* technique, it cannot use pretrained word representation vectors, because such word embeddings with WSD tokens do not exist. Instead, it requires training word embeddings from scratch, based on the corpus of available documents. The reason is that the generalization strategies discussed before (Section 4.3) produce a vocabulary that contains WSD identifiers (e.g., *bank.n.02*) and named entities (e.g., LOC, GPE, or ORG) and terms of this form are not typically included in the vocabularies of pretrained models.

5.2 Deep Learning Model

Summary creation is performed by a deep seq2seq model, based on RNNs (Lipton, Berkowitz, and Elkan 2015). More specifically, this model predicts an output sequence of tokens $Y' = (y'_1,\ y'_2, \ldots)$ (the summary), given an input sequence of tokens $X = (x_1,\ x_2, \ldots)$ (the text to be summarized).

In this work, we use five seq2seq models: (i) attentive, (ii) pointer-generator, (iii) reinforcement learning, (iv) transformer, and (v) pretrained encoder transformer. The particular neural networks are described in detail below.

5.2.1 Attentive Sequence-to-Sequence Model. Figure 3 illustrates the overall network, which is a seq2seq model of encoder-decoder architecture, along with an attention mechanism (See, Liu, and Manning 2017). The encoder takes the word embeddings of the original text, and the decoder learns to predict the respective summary. The specific components of the network are described in more detail below.

Embedding Layer: The embedding layer receives a word from the source text, maps it to the embedding space (vector representation) and subsequently forwards it to the next encoding layer. Word embeddings are formed from text-summary pairs during training, prior to becoming available to the embedding layer.

Bidirectional LSTM Layer (Encoder): The second layer of the encoder consists of bidirectional LSTM units (Graves, Jaitly, and Mohamed 2013; Lipton, Berkowitz, and Elkan 2015), which are provided with the word embeddings of a sequence of tokens of the source text $X = (x_1,\ x_2, \ldots,\ x_n)$ (one word-vector for each time step) in forward

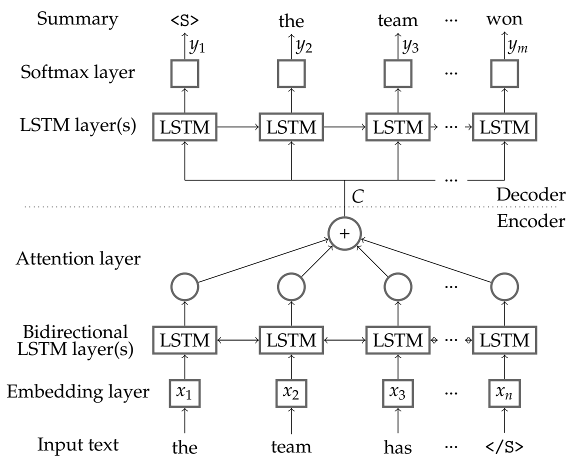


Figure 3
The deep learning model.

and reverse order (hence the bidirectional structure), producing a hidden state $H_{e,t}$ at their output. This hidden state is actually the concatenation of the hidden state vectors of both directions of the bidirectional LSTM.

Attention Layer: The last layer of the encoder consists of an attention mechanism (Bahdanau, Cho, and Bengio 2014; See, Liu, and Manning 2017) that focuses on relevant words of the input text, enhancing the accuracy of output predictions. This mechanism computes the context vector c_t, which is the weighted sum of the encoder hidden states $H_{e,t}$.

LSTM Layer (Decoder): The first layer of the decoder is comprised of unidirectional LSTM units. Their purpose is to predict the next word y_t in the summary, based on their hidden state $H_{d,t}$ at time t, the context vector c_t, and the previous hidden state $H_{d,t-1}$ of the decoder. During training, the target sequence of word vectors $Y = (y_1,\ y_2, \ldots,\ y_m)$ is also made available to the decoder (one-by-one word embedding of a reference summary, at each time step t) and the decoder learns to predict the next one, shaping the final summary.

Softmax Layer: The last layer of the decoder is the softmax layer, which generates the probability distribution of the next word over the set of candidate words. In particular, at each time step t, the softmax layer computes the probability of each candidate word y_i of the vocabulary Y for the predicted summary. Naturally, the sum of probabilities over the set of candidate words is equal to one.

The deep learning model outlined above is trained end-to-end via supervised learning on a corpus of text-summary pairs, using stochastic gradient descent and minimizing the negative log-likelihood of the target word y_t Equation (1):

$$Loss = -\sum_{t\,=\,1}^{T} logP(y_t|X) \tag{1}$$

where here $P(y_t|X)$ is the likelihood of the target word at time step t (of a set of T time steps which correspond to T words in the summary), given an input sequence of text X.

Additionally, and in order to avoid over-fitting, *dropout* is also used (Srivastava et al. 2014; Zaremba, Sutskever, and Vinyals 2014; Watt and du Plessis 2018)—this randomly drops connections of units from the neural network during training. Finally, the prediction of an optimal summary is further assessed through *beam search* (Graves 2012; Boulanger-Lewandowski, Bengio, and Vincent 2013). More specifically, at each time step of the beam search based decoder, the w candidate tokens of the highest log-probability are kept, in order for the *beam search algorithm* to determine the best output summary, where w is the beam-width parameter.

5.2.2 Pointer-Generator seq2seq Model. The pointer-generator (PG) network (See, Liu, and Manning 2017) extends the attentive seq2seq model of Section 5.2.1, by allowing OOV words to be copied from the source text to create the summary. This network generates words by either sampling them from a fixed vocabulary (i.e., the vocabulary of the training set) or from the words that appear in the source document. In other words, at each timestep, the network samples a word either from the vocabulary distribution of the training set or from the attention distribution. Therefore, the vocabulary is extended to include the OOV words of the source text for every document. It should be noted that the context vector of the attention mechanism is computed in a similar manner to the model of Section 5.2.1. Additionally, the PG seq2seq model incorporates a coverage mechanism that overcomes the problem of word repetition.

5.2.3 Reinforcement-Learning Model. In contrast to the models of sections 5.2.1 and 5.2.2 that minimize a maximum-likelihood loss, which is not used as an evaluation measure in TS, a reinforcement learning (RL) approach learns a policy that maximizes a particular metric, such as the ROUGE-L score. The utilized model extends a seq2seq point-generator network, by following the approach of Paulus, Xiong, and Socher (2018), incorporating the self-critical sequence training algorithm of Rennie et al. (2017). In particular, an encoder-decoder architecture (e.g., the PG model described above) is used as an agent, which learns to interact with a given environment, maximizing a reward. In our implementation, the reward function of the RL objective compares the target sequence with the respective ground truth sequence, in terms of the ROUGE-L metric. Additionally, the model incorporates pointer generation for dealing with the OOV words (e.g., words that remain OOV after the process of text generalization) and a coverage mechanism for avoiding the repetition problem.

5.2.4 Transformer-Based Models. Because transformer-based models (Vaswani et al. 2017; Zhang, Xu, and Wang 2019; You et al. 2019; Liu and Lapata 2019; Pilault et al. 2020; Xu et al. 2020) are the dominant approach in natural language understanding processes (Wolf et al. 2020), we have combined the proposed methodology with such architectures in an effort to investigate the effect of the present framework in transformer-based language models. In this direction, we follow the methodology of Liu and Lapata (2019) for abstractive text summarization, using two transformer-based approaches; (i) a transformer (TR) network and (ii) a pretrained encoder transformer (PETR) model. The TR model is a basic transformer language approach that uses self-attention layers in an encoder-decoder architecture, as presented by Vaswani et al. (2017). The second model uses a pretrained BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al. 2018) encoder with a transformer decoder, as described in Liu and Lapata (2019). The first model is trained from scratch, while the second model's weights are fine-tuned on the training sets of the employed data sets.

6. Post Processing

As has already been discussed (Sections 3 and 5), the machine learning methodology is applied on a generalized version of the input text and consequently the predicted summaries are also in a generalized form. This means that they contain general concepts such as hypernyms or named entities of the original terms. Therefore, a post-processing phase is necessary in order to replace the generalized concepts of the predicted summary with specific ones, shaping, in this way, the final summary in human-readable form. Algorithm 3 below performs this task, as it transforms a generalized summary to the final one, matching and replacing the general concepts of the predicted summary with the respective, specific, concepts of the original text.

Algorithm 3 Transforming the predicted summary (*predSum*) to its final, human-readable version

```
Require: $predSum$, $text$, $T$
 1: $cr \leftarrow \{\}$                          ▷ candidate replacements of generalized concepts
 2: $gc \leftarrow \{\}$                          ▷ generalized concepts
 3: for all $token_s \in predSum$ do
 4:     if $token_s$ is generalized then
 5:         $gc \leftarrow gc \cup \{token_s\}$
 6:         $w_s \leftarrow$ window text around $token_s$
 7:         for all $token_t \in text$ do
 8:             $w_t \leftarrow$ window text around $token_t$
 9:             $sim \leftarrow$ Similarity between $w_s$ and $w_t$ (Algorithm 4 or 5)
10:             $cr \leftarrow cr \cup \{(token_s, token_t, sim)\}$
11:         end for
12:     end if
13: end for
14: $summary \leftarrow$ matching the generalized concepts of $predSum$ to specific ones (optimal
    matching of Subsection 6.2.1 or greedy matching of Subsection 6.2.2)
15: return $summary$
```

The inputs to Algorithm 3 are the predicted summary *predSum*, the respective original *text*, and a taxonomy of concepts T. In the beginning, the set of candidate replacements of generalized concepts *cr* and the set of generalized concepts *gc* are initialized to empty sets (lines 1–2). Then, each token of the summary ($token_s$) is accessed (lines 3–13) and if it is in a generalized form (lines 4–12), then it is added to *gc* and the window of text around the $token_s$ is assigned to w_s (lines 5–6). Subsequently, each token of the original *text* ($token_t$) is accessed (lines 7–11), the text window around a $token_t$ is assigned to w_t (line 8), and the similarity *sim* between w_s and w_t is estimated, according to either Algorithm 4 or 5 (line 9), with the tuple of ($token_s$, $token_t$, sim) being added to *cr*. Finally, and prior to returning the human-readable *summary* (line 15), Algorithm 3 matches the generalized concepts in *perdSum* to specific ones, using optimal concept matching (Section 6.2.1) or the greedy approach of Algorithm 7 (Section 6.2.2).

Identifying Generalized Concepts. At line 4 of Algorithm 3, it is determined whether a token of the input summary is generalized or not, by identifying those concepts that carry a relative identifier. For example, in NEG, the generalized words are named entities (of the form PERSON, LOC, ORG, etc.), whereas in LG-based approaches, they have been changed to the respective WSD identifier (e.g., *tree.n.01*, *bank.n.02*). Additionally, in cases of combined approaches like NEG-LG, the generalized concepts are identified accordingly.

6.1 Text Similarity

Text similarity between excerpts of the generalized summary and the original text (line 9, Algorithm 4) is an integral part of the post-processing phase, because it determines which specific concept fits best to a general one. For this task, Algorithms 4 and 5 are proposed for LG-based models (LG or W-LG) and NEG, respectively.

Algorithm 4 Text similarity based on LG

```
Require: $token_s$, $token_t$, $token_g$, $w_s$, $w_t$, $T$, $a_1$, $a_2$, $a_3$
 1: $hyponyms \leftarrow$ hyponyms of $token_s$
 2: $hypernyms \leftarrow$ hypernyms of $token_s$
 3: $sim \leftarrow similarity\,((token_s, w_s),\ (token_t, w_t))$
 4: if $token_s \equiv\ token_g$ then
 5:     $sim = a_1\ \cdot\ sim$
 6: else if $token_t \in hyponyms$ then
 7:     $sim = a_2\ \cdot\ sim$
 8: else if $token_g \in hypernyms$ then
 9:     $sim = a_3\ \cdot\ sim$
10: end if
11: return $sim$
```

LG-Based Similarity. Algorithm 4 estimates the similarity between concepts for LG-based models (LG or W-LG). Its input includes a token of the generalized summary ($token_s$), a token of the generalized text ($token_g$), a text window for the generalized summary (w_s), a text window for the original text (w_t), a taxonomy of concepts T, and three hyperparameters a_1, a_2, a_3, which regulate the significance of the similarity between w_s and w_t and whose optimal values are determined experimentally. Initially, the taxonomy paths of hyponyms and hypernyms of $token_s$ are retrieved (lines 1–2). Then, the similarity between $token_s$ and $token_g$ is computed and stored in sim (line 3), for text windows of sizes w_s and w_t, respectively, according to the selected similarity function (e.g. cosine similarity or word-mover-distance-based similarity). In case $token_s$ is equal to the $token_g$ (line 4), that is, when the generalized token of *genSum* is equal to the token of the generalized text, sim is further amplified by a factor a_1; otherwise, if $token_g$ is in the hyponym or hypernym path of $token_s$, sim is enhanced by coefficients a_2 or a_3, respectively (lines 6–10). Finally, the algorithm returns the similarity sim between the two text excerpts (line 11).

NEG-Based Similarity. Algorithm 5 estimates text similarity for NEG. Its input is a token ($token_s$) and a window (w_s) of the generalized summary, a token ($token_t$) and a window (w_t) of the original text, a token of the generalized text ($token_g$), a taxonomy of concepts T, and hyperparameter b. In the beginning, the similarity between text windows w_s, w_t of $token_s$, $token_a$ is computed (line 1) according to the utilized similarity function (which can be any of those discussed in the previous case). If $token_s$ is equal to the token of the generalized article $token_g$, the initial similarity is multiplied by b (line 3). Finally, the algorithm returns the similarity sim between the two excerpts (line 5).

Algorithm 5 Text similarity based on NEG

```
Require: $token_s$, $token_t$, $token_g$, $w_s$, $w_t$, $T$, $b$
1: $sim \leftarrow similarity\,((token_s, w_s),\ (token_t, w_t))$
2: if $token_s \equiv\ token_g$ then
3:     $sim = b\ \cdot\ sim$
4: end if
5: return $sim$
```

Similarity in W-NEG, NEG-LG, and W-NEG-LG. In these cases, the final summary is produced after applying Algorithm 3 twice; firstly, the generalized summary is transformed into a mid-summary using Algorithm 4 at the similarity step. Then, the produced mid-summary is once again provided as input to Algorithm 3, using Algorithm 5 as the similarity function this time, in order to produce the final summary.

6.2 Concept Matching

Algorithm 3 also requires a procedure to replace the generalized concepts of the predicted summary with specific ones, in order to produce the final summary in human-readable form (line 14). This task is formulated in a graph-theoretical context, as a bipartite matching problem among general concepts (first set of nodes) and their specific counterparts (second set of nodes). To this end, two techniques are described below; the *optimal* (Section 6.2.1) and the *greedy* (Section 6.2.2).

6.2.1 Optimal Concept Matching. This approach reduces optimal bipartite matching to a minimum cost flow problem (Hansen and Klopfer 2006; Dasgupta, Papadimitriou, and Vazirani 2008; Brunsch et al. 2013). More specifically, the minimum cost flow graph G (as, for example, in Figure 4) consists of a source node Src, the set of nodes of the general concepts N_G (i.e., concepts of the generalized summary), the set of nodes of the candidate concepts N_C (i.e., concepts which are included in the original document), and a terminal node Trm. Also, the nodes are connected with edges carrying the information

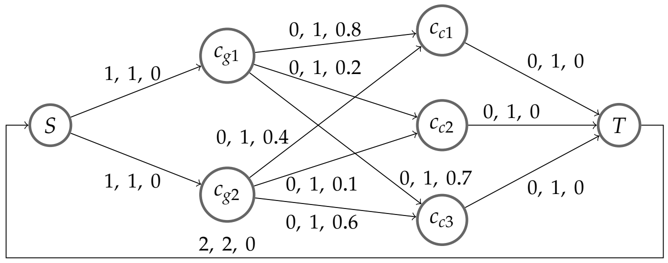


Figure 4
A simple example of a minimum cost flow graph for optimal matching among general concepts c_{gi} and candidate concepts c_{cj}, with edges carrying min_f, max_f, and $cost$.

of a minimum (min_f) and a maximum (max_f) flow, along with a flow $cost_{i,j}$ value, given in Equation (2):

$$cost_{ij} = \begin{cases} M - similarity(i,\ j), & i \in N_G \text{ and } j \in N_C \\ 0, & i = Src \text{ or } j = Trm \end{cases} \tag{2}$$

where M denotes the maximum similarity score and $similarity(i,\ j)$ is the similarity between a text-window around the general concept of node $i \in N_G$ (i.e., text-window of the generalized summary) and a text-window around the candidate concept of node $j \in N_C$ (i.e., text-window of the original text).

The constraints min_f, max_f between nodes i, j are expressed in the form of a tuple Equation (3). The flow among Src and N_G is equal to 1, as all the generalized concepts need to be matched with candidate ones. Additionally, the flow among nodes in N_G and nodes in N_C and, in turn, the flow among nodes in N_C and Trm is equal to 0 or 1, in order to express the restriction that each general concept can be matched with only one candidate concept. Finally, the flow from Src to Trm is equal to the number of general concepts $|N_G|$ that need to be matched with candidate ones in N_C.

$$(min_{f_{ij}},\ max_{f_{ij}}) = \begin{cases} (1,\ 1), & i = Src,\ j \in N_G \\ (0,\ 1), & i \in N_G,\ j \in N_C \\ (0,\ 1), & i \in N_C,\ j = Trm \\ (|N_G|,\ |N_G|), & i = Src,\ j = Trm \end{cases} \tag{3}$$

In the general case, the minimum cost flow problem may be re-formulated in terms of integer linear programming (Bertsekas 1998), as expressed in Equation (4), where $V = N_C \cup N_G \cup \{Src,\ Trc\}$ is the overall set of nodes.

$$\begin{aligned} \text{minimize:} \quad & \sum_{i,j \in V, i \neq j} cost_{ij} \cdot f_{ij} \\ \text{subject to:} \quad & min_{f_{ij}} \leq f_{ij} \leq max_{f_{ij}}, \forall i,j \in V,\ i \neq j \\ & \sum_{j \in V, i \neq j} f_{ij} = 0,\ \forall i \in V \\ & f_{ij} \in \mathbb{Z}_{\geq 0} \end{aligned} \tag{4}$$

Optimal concept matching is further illustrated in Algorithm 6 and Example 8.

Algorithm 6 describes the procedure of performing the optimal concept matching among the generalized concepts c_g of the predicted summary and the candidate concepts c_c of the original text. The input to the algorithm is a generalized summary *genSum* and a set of candidate replacements *cr*, which contains tuples of a general concept, a candidate concept, and their similarity. First, the flow graph is created (line 1) and the minimum cost flow in the graph is determined (line 2; e.g., by linear programming). Then, edges with flow $f_{ij} > 0$ correspond to concepts that match (line 3), and the tuples of these concepts are assigned to the set of *replacements* (line 4). Then, *summary* is initialized to the generalized summary (line 5) and for each tuple in

Algorithm 6 Optimal concept matching

```
Require: genSum, cr
1: The flow graph is created according to cr
2: The minimum cost flow problem is solved
3: The edges with flow $f_{ij} > 0$ correspond to concept that matches
4: replacements ← tuples of concepts that matches
5: summary ← genSum
6: for all $(c_{gi}, c_{cj}) \in$ replacements do
7:     summary ← replace $c_{gi}$ with $c_{cj}$
8: end for
9: return summary
```

the *replacements* set, the generalized concepts of the *summary* are replaced with candidate ones (lines 6–8). In the end, the algorithm returns the final *summary* (line 9).

Example 8 (Optimal Concept Matching)

Let us assume that a predicted summary includes general concepts that need to be matched with specific ones in the original text. In the post-processing phase, after measuring the similarity among concepts, the set of candidate replacements *cr* of Algorithm 3 is as follows

$$cr = \{(c_{g1}, c_{c1}, 0.2), (c_{g1}, c_{c2}, 0.8), (c_{g1}, c_{c3}, 0.3),$$
$$(c_{g2}, c_{c1}, 0.6), (c_{g2}, c_{c2}, 0.9), (c_{g2}, c_{c3}, 0.4),\}$$

where two generalized concepts $c_{gi} \in N_G$, $i = \{1, 2\}$ and three candidate concepts $c_{cj} \in N_C$, $j = \{1, 2, 3\}$ exist. Figure 4 depicts the corresponding flow graph, where the edges carry the constraints of minimum/maximum flow and the cost (min_f, max_f, and *cost*). The similarity score is assumed to take values in $[0, 1]$ and it is transformed into cost, according to Equation (2), with $M = 1$.

The optimal matching in this case is that of the pairs of concepts $c_{g1} - c_{c2}$ and $c_{g2} - c_{c1}$, which minimize the flow cost, obtaining a minimum cost of 0.6 (i.e., 0.2 + 0.4). Therefore, concepts c_{g1}, c_{g2} of the generalized summary will be replaced by c_{c2}, c_{c1}, respectively.

6.2.2 Greedy Concept Matching. The optimal concept matching approach discussed above exhibits high computational complexity (Hansen and Klopfer 2006; Kovács 2015) and requires an efficient linear programming solver. For this reason, a greedy algorithm is also proposed, which has lower computational complexity. Algorithm 7, which matches the generalized concepts of the predicted summary to the candidate concepts of the original text, requires a generalized summary *genSum*, a list of candidate replacements *cr*, and a list of generalized concepts *gc*. Initially, *cr*, is sorted in descending order according to similarity *sim* (line 1) and the generalized summary *genSum* is assigned to the final *summary* (line 2). Then, the tuples of *cr* (line 3), which contain a generalized word ($token_s$) of the general summary, a word ($token_t$) of the original text and their similarity score *sim*, are examined in descending (similarity) order (lines 3–11). If $token_s$ belongs to *gc* (line 4), it is replaced in the *summary* by $token_t$ (line 5) and is subsequently removed from *gc* (line 6). Once all generalized concepts in *gc* have been replaced by specific ones, or all candidate replacements in *cr* have been examined, the algorithm terminates, returning the final summary (line 12).

Algorithm 7 Greedy concept matching

```
Require: $genSum$, $cr$, $gc$
 1: sort $cr$ in descending order of $sim$
 2: $summary \leftarrow genSum$
 3: for all $(token_s, token_t, sim) \in cr$ do
 4:     if $token_s \in gc$ then
 5:         $summary \leftarrow$ replace $token_s$ with $token_t$
 6:         $gc \leftarrow gc \setminus token_s$
 7:     end if
 8:     if $gc \equiv \{\}$ then
 9:         break for-loop
10:     end if
11: end for
12: return $summary$
```

Example 9 (Greedy Concept Matching)
Given the same set of candidate replacements *cr* of Example 8, the greedy algorithm, after sorting the tuples *cr* in descending order according to their similarity sore, matches the pairs of concepts as follows: $c_{g2} - c_{c2}$ and $c_{g1} - c_{c3}$, as the first pair has a higher similarity score than the second.

From Examples 8 and 9 presented above, it is evident the greedy matching might be different from the optimal, as the former is based on an approximate algorithm. Despite their differences, both approaches exhibit similar performance because their efficiency is primarily dependent on the chosen similarity function, as it is going to be analyzed in the forthcoming sections.

6.3 Computational Complexity

The computational complexity of Algorithm 3 is greatly influenced by the concept matching step on line 14. If optimal concept matching is used (Section 6.2.1), which is expressed in terms of integer linear programming, then it becomes a NP-complete problem [Matousek and Gärtner 2007]. Nevertheless, in the examined setting, where the number of generalized and candidate concepts are limited, it is feasible to find an optimal solution in reasonable time.

Instead of using optimal concept matching, the greedy solution (Algorithm 7, which is further described in Section 6.2.2) constitutes a viable alternative. In particular, assuming that the number of elements of the set of candidate replacements *cr* is equal to n (i.e., $n = |cr|$), the computational complexity, in the worst case, of sorting *cr* is $O(n \log n)$ (e.g., using heap sort) [Mishra and Garg 2008] and the complexity of the for-loop (lines 3–11) is linear ($O(n)$). Therefore, the overall computational complexity of Algorithm 7 is $O(n \log n)$.

Having defined the computational complexity of both concept matching methods (optimal and greedy), we may proceed in determining the complexity of the overall post-processing task (Algorithm 3). The nested for-loop of lines 7–11 has an $O(n^2)$

complexity, when all n input tokens of the predicted summary are examined. Regarding the concept matching step on line 14, if the optimal solution is used, then Algorithm 3 becomes NP-complete. Otherwise, if the greedy approach is chosen, Algorithm 3 runs in $O(n^2)$ time. In practice, the impact of the post-processing task on system performance becomes apparent only in cases of too many instances.

7. Experiments

The experiments presented in this section aim at investigating various aspects of the proposed methodology and at evaluating the overall framework, following experimental procedures that have been extensively adopted in similar works (Rush, Chopra, and Weston 2015; Nallapati et al. 2016; Chopra, Auli, and Rush 2016; See, Liu, and Manning 2017; Gao et al. 2020). Section 7.1 provides an overview of the used data sets, Section 7.2 contains a brief description of the evaluation metrics, Section 7.3 discusses the experimental procedure and hyperparameter tuning, and finally Section 7.4 examines the performance of other approaches in the literature, both baseline and state-of-the-art, on the same data sets.

7.1 Data Sets

The proposed framework has been evaluated on three popular summarization data sets; the annotated `Gigaword` (Napoles, Gormley, and Van Durme 2012), the `DUC 2004` (Over, Dang, and Harman 2007), and the `CNN/DailyMail` (Hermann et al. 2015). The specific version of the `Gigaword` data set used is that of Rush, Chopra, and Weston (2015), which has been widely adopted in the relevant literature (Joshi, Fernández, and Alegre 2018); it contains about 3.8 million article-summary pairs and a total of 123 million tokens, which form a vocabulary of 119 distinct tokens. The average length of an article is 31.4 tokens and the average length of a summary is 8.3 tokens. Out of the whole data set, 2,000 pairs have been randomly sampled to form the test set and another 2,000 have been similarly selected for the creation of the validation set; a common practice when this data set is used in the experiments (Rush, Chopra, and Weston 2015; Nallapati et al. 2016; See, Liu, and Manning 2017).

The second data set, `DUC 2004`, contains 500 news articles along with four human-generated reference summaries for each of them. It has also been pre-processed, retaining only the first sentence of the articles and reducing the summaries to a maximum length of 75 characters, as in Rush, Chopra, and Weston (2015), Chopra, Auli, and Rush (2016), and Gao et al. (2020). Because `DUC` contains very few instances for training a deep learning model, it is solely used for evaluation purposes (Gao et al. 2020; Chopra, Auli, and Rush 2016).

The last data set, `CNN/DailyMail`, is also widely used in document-level abstractive TS and it is composed of articles/stories and their multi-sentence summaries. A non-anonymized version has been obtained, after following the preprocessing steps proposed by the data set creators (Nallapati et al. 2016). After preprocessing, the data set contains 287,227 training article-summary pairs, 11,490 testing instances, and 13,368 validation samples. The articles have been limited to 400 tokens and the summaries to 100 tokens, following the typical procedure when this data set is used in deep learning models (See, Liu, and Manning 2017; Shi et al. 2018; Cohan et al. 2018). After limiting the source and target text, the average input and output lengths are 386.42 and 61.08 tokens, respectively.

Table 1
Distribution of nouns and verbs in data sets.

	Nouns		Verbs	
Gigaword (train set)	46,989,593	69.77%	20,362,923	30.23%
Gigaword (test set)	24,936	69.70%	10,840	30.30%
Gigaword (validation set)	24,883	70.23%	10,548	29.77%
DUC 2004	11,023	65.57%	5,787	34.43%
CNN/DailyMail (train set)	29,165,327	59.73%	19,665,826	40.27%
CNN/DailyMail (test set)	960,193	62.39%	578,803	37.61%
CNN/DailyMail (validation set)	1,121,401	62.54%	671,765	37.46%

Table 1 presents the noun and verb distributions on data sets, according to the utilized **part-of-speech** (POS) tagging process, because the proposed framework generalizes these specific lexical categories. It is evident that the majority of the detected POS are nouns.

Finally, it should be noted that, unlike our earlier work (Kouris, Alexandridis, and Stafylopatis 2019) described in Section 2, which further pre-processed the data sets in an effort to improve system performance (by removing duplicate entries, punctuation, too-long summaries, etc.), the present work retains the form of data sets as they are used by the competitive approaches (Tables 3 and 4). Therefore, a direct comparison among the experimental results of this framework and of other approaches is possible.

7.2 Evaluation Metrics

The evaluation of the proposed framework is based on the official *ROUGE* package (Lin 2004) and, more specifically, on the average score of *ROUGE-1* (word overlap), *ROUGE-2* (bigram overlap), and *ROUGE-L* (longest common sequence - LCS). In particular, the *F-1 score* of all three ROUGE metrics is calculated on the Gigaword and CNN/DailyMail data set, whereas in the case of DUC 2004, the *Recall* of the same metrics is reported, as is the practice in evaluating those data sets in literature (Nallapati et al. 2016; Chopra, Auli, and Rush 2016; Gao et al. 2020).

Additionally, to evaluate the factual consistency (Kryściński et al. 2019; Goodrich et al. 2019) of generated text, we follow the approach of Goodrich et al. (2019). In particular, facts, represented as *(subset, relation, object)* triplets, are retrieved from text. The factual accuracy $fact_{acc}$ is defined as the precision between the retrieved facts from the generated summary and those of the source text Equation (5):

$$fact_{acc} = \frac{|F_t \cap F_g|}{|F_g|} \tag{5}$$

where F_t is a set of facts from the source text and F_g is a set of facts of generated text. The fact triplets have been retrieved according to the open information extraction *(OpenIE)* approach (Angeli, Premkumar, and Manning 2015; Goodrich et al. 2019). Factual consistency is assessed by determining the extend to which facts of the generated summaries cover the facts of the source text.

Table 2
NER tags used in NEG-based strategies.

PERSON (people)	LAW (Named documents made into law)
NORP (nationalities, religious, groups, etc.)	LANGUAGE (named languages)
FAC (facilities, buildings, airports, etc.)	DATE (dates or periods)
ORG (organizations, companies, etc.)	TIME (times smaller than a day)
GPE (countries, cities, states)	PERCENT (percentage)
LOC (non GPE locations)	MONEY (monetary values, including unit)
PRODUCT (objects, vehicles, foods, etc.)	QUANTITY (weight, distance, etc.)
EVENT (named hurricanes, sport events, etc.)	ORDINAL (first, second, etc.)
WORK_OF_ART (books, films, etc.)	CARDINAL (numerals, none of other types)

7.3 Experimental Procedure and Parameter Tuning

Data Pre-processing. The generalization methodology of Section 4.2 is applied to the training data, in order to examine its effect on the proposed methodology. In particular, the LG, NEG, and NEG-LG strategies are considered (Section 4.3), along with their WSD-based versions (W-LG, W-NEG, and W-NEG-LG). For each of these strategies, thresholds θ_f, θ_d are set to various levels, in order to study their influence on the obtained results. Finally, the taxonomy of concepts used is that of *WordNet* (Miller 1995; Fellbaum 1998).

Content generalization is applied to (i) nouns and (ii) both nouns and verbs using POS tagging.[1] Additionally, NER[2] is invoked by the NEG-based strategies. Table 2 outlines the entity types of the NER task, which uses a pretrained model on the `OntoNotes 5.0` data set (Weischedel et al. 2013). Natural language processing on the POS and NER tags is based on modern algorithms of proven performance (Honnibal and Johnson 2015; Choi, Tetreault, and Stent 2015; Andor et al. 2016). Additionally, the knowledge-based WSD used is in line with the utilized taxonomy of concepts (*WordNet*). The WSD identifiers specify the sense of its synsets (e.g., *play.n.01*), using the adaptive Lesk algorithm (Banerjee and Pedersen 2002, 2003).

As a result of the process described above, several versions of text-summary pairs are created, to be used in training and evaluation. The particular generalization schemes and the performance of the corresponding models are further discussed in Section 7.

Word Embeddings. The *word2vec* vector-based representation is used to obtain word embeddings of dimensionality 300. More specifically, a *word2vec* model of continuous bag-of-words architecture (Mikolov et al. 2013) is trained for each version of the training data. The window size has been set to 5 and each model has been trained for 10 epochs with a decaying learning rate from 0.025 to 0.001.

Training the Attentive seq2seq Model. For the different versions of the training data, an equal number of deep learning models has been trained (Section 5.2), with their parameters having been optimized on the validation set of `Gigaword`. The bidirectional LSTM layer of the encoder consists of two layers of equal size each (200), which is also the size of the LSTM layer of the decoder. The batch size is set to 64 for the `Gigaword` and 32 for the CNN/DailyMail data sets, while the training data for each epoch are

1 Spacy POS tagger implementation: `https://spacy.io/usage/linguistic-features#pos-tagging`.
2 Spacy NER implementation: `https://spacy.io/usage/linguistic-features#namedentities`.

randomly shuffled. The learning rate is initially set to 0.002, decaying by a factor of 25% on each training epoch. Additionally, the *Adam* optimization method (Kingma and Ba 2014) is used, with gradient norm clipping (Pascanu, Mikolov, and Bengio 2013) and negative conditional log-likelihood as the loss function (Golik, Doetsch, and Ney 2013). Finally, *dropout* with $p = 0.2$ is also used. In the case of the CNN/DailyMail data sets, the vocabulary is limited to 150,000 words (i.e., using the most frequent tokens of training set) while the `Gigaword` data set is used without limitations. Model training took place on NVidia K40 GPUs and the models converged sufficiently after 15 epochs.

Training the PG Model. The differences between the PG model (presented in Section 5.2.2) and the attentive seq2seq model described above is that the encoder consists of two layers of bidirectional LSTMs of 256 dimensions, while the decoder uses one LSTM of dimensionality 512.

Training the RL Model (Section 5.2.3). The learning rate is fixed to 10^{-4} and the batch size is equal to 32 and 16 for the `Gigaword` and CNN/DailyMail data sets, respectively. The dimensionality of the LSTM layer is similar to the PG model mentioned above. For the rest of the training parameters, the same assumptions with the above-described architectures are made.

Training the TR Model (Section 5.2.4). The encoder and decoder consist of a stack of 6 layers each. The model dimensionality is equal to 512 and the inner-layer dimensionality has been set to 2,048. We assume 8 heads (i.e., 8 parallel attention layers, which reduces the model dimensionality of each attention layer to 512/8 = 64). The Adam optimizer is used with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The learning rate is adjusted during training according to Equation (6) (i.e., increasing the learning rate for the first *warmupSteps* training steps and then decreasing it), where *warmupSteps* $= 5{,}000$ and $a = 0.05$.

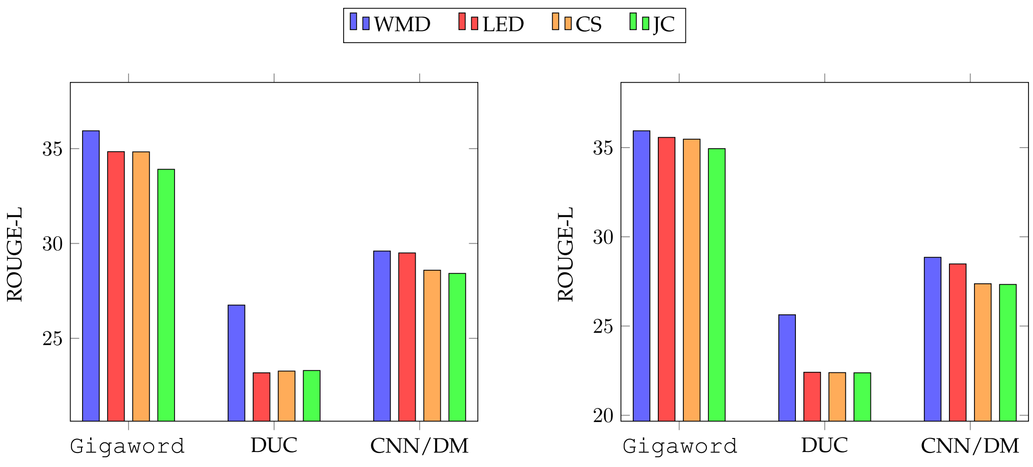
$$lr = a \cdot \min\{step^{-0.5},\ step \cdot warmupSteps^{-1.5}\} \tag{6}$$

Dropout is applied with probability $p = 0.1$ and batch size is set to 64 and 16 for the `Gigaword` and CNN/DailyMail data sets, respectively.

Training the PETR Model (Section 5.2.4). The difference between the TR and PETR models is that the latter uses a pretrained BERT encoder and a transformer decoder of 6 layers, which are initialized randomly. Moreover, for a stable fine-tuning, two Adam optimizers are used, separately for the encoder and the decoder. Each optimizer (with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$) assumes different learning rates according to Equation (6), where $a_{enc} = 10^{-3}$, $a_{dec} = 0.1$, $warmupSteps_{enc} = 10{,}000$, and $warmupSteps_{dec} = 5,000$. The different learning rates aim at a stable model hyperparameter tuning, where the pretrained encoder will be fine-tuned with a smaller learning rate and smoother decay compared to the decoder (i.e., avoiding encoder/decoder overfitting/underfitting) (Liu and Lapata 2019).

Generating System Summaries. Beam search of size 4 has been used at evaluation, to optimize the output summary.

Post-processing. Post-processing has been extensively described in Section 6. For the `Gigaword` data set, the optimal value of coefficients a_1, a_2, and a_3 of Algorithm 4 have been determined to be 2.0, 1.5, and 1.5, respectively, after thorough experimentation, while coefficient b of Algorithm 5 has been similarly set to 2.0. Additionally, a range of text similarity metrics for Algorithms 4 and 5 has been considered, like *cosine similarity* (CS), *Jaccard coefficient* (JC), *word mover distance* (WMD) (Kusner et al. 2015), and the *Levenshtein edit distance* (LED) (Yujian and Bo 2007). WMD, which utilizes word embeddings, taking into account syntactic and semantic aspects of the text, obtained



(a) LG strategy ($\theta_f = 200$ and $\theta_d = 5$) (b) NEG strategy ($\theta_f = 200$)

Figure 5
ROUGE-L F1 scores for the various text similarity metrics (WMD, LED, CS, JC) considered in the post-processing phase for concept matching.

the best results (after converting the distance to similarity). In particular, the optimal performance has been achieved when the context windows around the candidate and the generalized concepts were set to 10 and 6, respectively. More specifically, the optimal values of the post-processing parameters have been determined through exhaustive grid search (maximizing the ROUGE-L F1 score) on the `Gigaword` validation set. The same hyperparameter values have been retained in `DUC` and `CNN/DailyMail` data sets. Figure 5 illustrates the ROUGE-L F1 scores for the aforementioned text similarity metrics on the LG and NEG strategies; WMD tends to outperform the other approaches, especially in the `DUC` data set. Finally, both approaches, optimal and greedy (sections 6.2.1 and 6.2.2), have been used for concept matching, exhibiting almost the same performance in terms of the ROUGE scores. For this reason, the optimal concept matching technique has been selected in the final experiments, in an effort to validate whether the reported ROUGE scores are the maximum to be obtained. Additionally, to indicate the differences between the greedy and optimal concept matching methods, the results of the greedy matching algorithm are reported in Section 8.1.

7.4 Baseline Approaches

The baseline approach is composed of the proposed deep learning models (Section 5.2), with the optimized hyperparameter values presented in the previous subsection, but without applying any generalization strategy or post-processing task on the training data. Additionally, Tables 3 and 4 summarize the performance of other, state-of-the-art, approaches on the same data sets, as reported by their respective authors (dashes in Table 3 indicate that they did not consider the specific data set and/or metric in their experiments). Because the authors of the other approaches have obtained evaluation and test sets of the same size as ours and with the same methodology (random sampling with replacement) on the `Gigaword` and `CNN/DailyMail` data set and have also used the `DUC 2004` data set for evaluation purposes only, the performance results are directly comparable. An extra metric reported for some models on the `Gigaword` data set (5th

Table 3
ROUGE and NTR scores on Gigaword and DUC 2004 data sets of other, state-of-the-art approaches.

Approach	Gigaword				DUC 2004		
	ROUGE-1	**ROUGE-2**	**ROUGE-L**	**NTR %**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
ABS+ (Rush, Chopra, and Weston 2015)	31.00	12.65	28.34	8.50	28.18	8.49	23.81
RAS-Elman (Chopra, Auli, and Rush 2016)	33.78	15.97	31.15	–	28.97	8.26	24.06
Words-lvt2k-1sent (Nallapati et al. 2016)	34.97	17.17	32.70	24.15	28.35	9.46	24.59
Words-lvt5k-1sent (Nallapati et al. 2016)	35.30	16.64	32.62	–	28.61	9.42	25.24
Model #8 (Nallapati, Xiang, and Zhou 2016)	35.30	17.58	32.88	22.89	–	–	–
+CGU (Lin et al. 2018)	36.30	18.00	33.80	–	–	–	–
GLEAM (Gao et al. 2020)	36.51	16.80	33.92	–	29.51	9.78	25.60
Beam+BPNorm (Song et al. 2020)	39.19	20.38	36.69	–	–	–	–
Prophnet (Yan et al. 2020)	39.55	20.27	36.57	–	–	–	–
SAGCopy-Indegree-1 (Xu et al. 2020)	38.84	20.39	36.27	–	–	–	–

Table 4
ROUGE scores on CNN/DailyMail data set of other, state-of-the-art approaches.

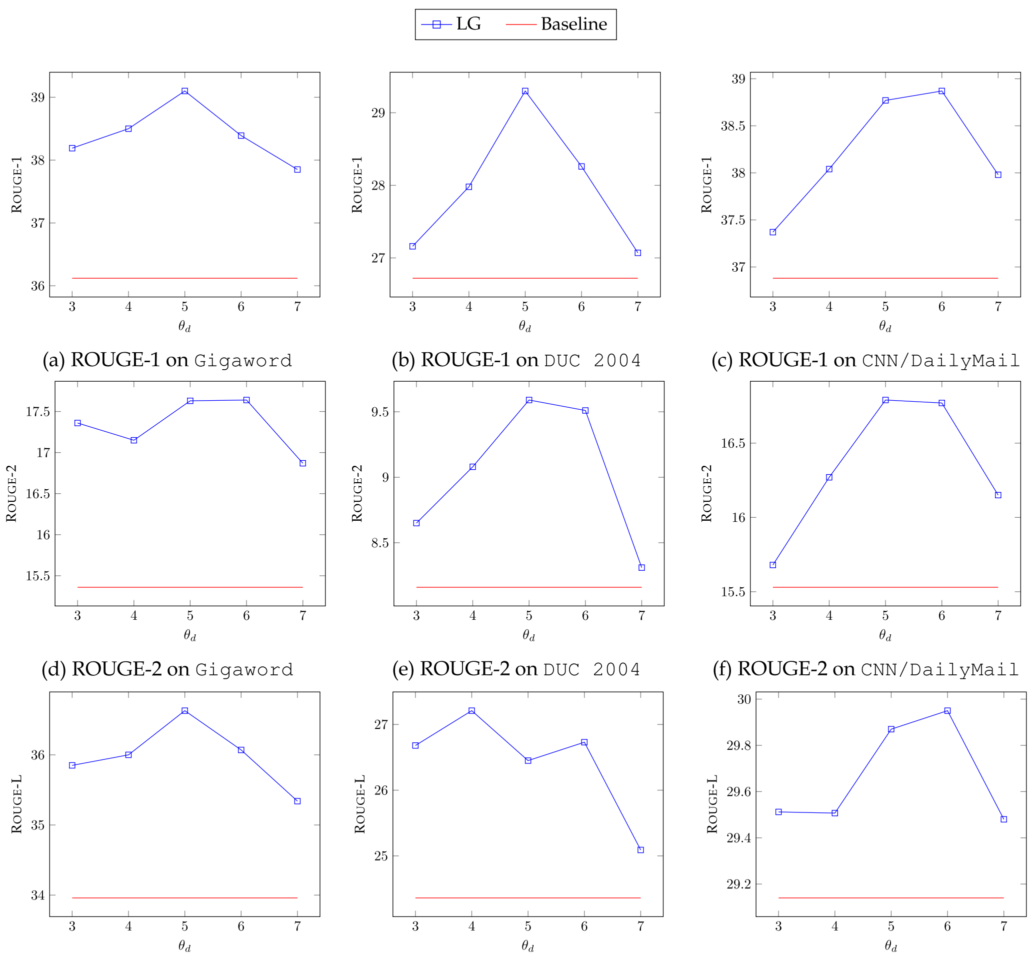
Approach	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
words-lvt2k-temp-att (Nallapati et al. 2016)	35.46	13.30	32.65
ML, with intra-attention (Paulus, Xiong, and Socher 2018)	38.30	14.81	35.49
RL, with intra-attention (Paulus, Xiong, and Socher 2018)	41.16	15.75	39.08
pointer-generator + coverage (See, Liu, and Manning 2017)	39.53	17.28	36.38
rnn-ext+abs+RL+rerank (Chen and Bansal 2018)	40.88	17.80	38.54
Bottom-Up Summarization (Gehrmann, Deng, and Rush 2018)	41.22	18.68	38.34
ROUGESal+Ent (Pasunuru and Bansal 2018)	40.43	18.00	37.10
SENECA (Sharma et al. 2019)	41.52	18.36	38.09
BertSumAbs (Liu and Lapata 2019)	41.72	19.39	38.76
ETADS (You et al. 2019)	41.75	19.01	38.89
Two-Stage+RL (Zhang, Xu, and Wang 2019)	41.71	19.49	38.79
ProphetNet (Yan et al. 2020)	43.68	20.64	40.72
SAGCopy-Outdegree (Xu et al. 2020)	42.53	19.92	39.44

column of Table 3) is that of *New Tokens Rate* (NTR), which is the percentage of tokens appearing in the generated summary, but not included in the input text.

8. Results

Initially, the effect of the level of generalization on the performance of TS is investigated. Figure 6 depicts ROUGE-1, ROUGE-2, and ROUGE-L scores on the Gigaword (Figures 6a, 6d, 6g), DUC 2004 (Figures 6b, 6e, 6h), and CNN/DailyMail (Figures 6c, 6f, 6i) data sets and for various levels of generalization, as controlled by threshold θ_d of the LG strategy ($\theta_d = \{3, 4, 5, 6, 7\}$). The word frequency threshold of candidate concepts for generalization has been kept constant at $\theta_f = 100$. Because the efficiency (in terms of ROUGE score) is maximized for $\theta_d = 5$ for two of three data sets, it is fixed to this value for the rest of the experiments.

Table 5 illustrates the ROUGE scores on data sets for LG, NEG, and NEG-LG and for varying thresholds of minimum word frequency for generalization ($\theta_f = \{100, 200, 500, 1{,}000\}$). The last row of this table outlines the performance of the baseline model (Section 7.4). In this set of experiments, noun generalization is assumed with the



(a) ROUGE-1 on `Gigaword` (b) ROUGE-1 on `DUC 2004` (c) ROUGE-1 on `CNN/DailyMail`

(d) ROUGE-2 on `Gigaword` (e) ROUGE-2 on `DUC 2004` (f) ROUGE-2 on `CNN/DailyMail`

(g) ROUGE-L on `Gigaword` (h) ROUGE-L on `DUC 2004` (i) ROUGE-L on `CNN/DailyMail`

Figure 6
ROUGE scores on data sets for varying levels of generalization (θ_d) for the LG strategy ($\theta_f = 100$).

minimum taxonomy depth fixed to 5, as described above. Additionally, NTR percentage is reported for the `Gigaword` data set. All examined models outperform the baseline and the majority of them achieve higher ROUGE scores than the state-of-the-art approaches (Table 3).

The statistical significance of the ROUGE scores, reported for the various strategies, is assessed by Welch's *t*-test (Sakai 2016). Using the ROUGE scores for each sample of the test set (for every data set), the *t*-test is performed on each pair of models for every ROUGE metric. The obtained significance values are reported in the captions of Tables 5–10 and prove that the differences in ROUGE scores are statistically significant in all of the examined cases.

Table 6 summarizes the same metrics for the WSD-based noun generalization for varying θ_f. In particular, the utilized strategies are those of W-LG, W-NEG, and W-NEG-LG, which apply LG, NEG, and NEG-LG on a disambiguated version of the original text. Again, the last row of this table exhibits the performance of WSD-n, which is assumed to be the baseline model in this case. WSD-n is based on a WSD version of

Table 5
ROUGE and NTR scores for: (i) LG, NEG, and NEG-LG strategies, (ii) varying θ_f ($\theta_d = 5$), and (iii) nouns only ($p_{value} < 0.012$ for ROUGE-1, $p_{value} < 0.02$ for ROUGE-2 and ROUGE-L).

Model	θ_f	Gigaword				DUC 2004			CNN/DailyMail		
		R-1	**R-2**	**R-L**	**NTR %**	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
LG-f100-d5-n	100	39.09	17.63	36.63	24.97	29.30	9.59	26.68	38.77	16.79	29.87
LG-f200-d5-n	200	38.23	17.29	35.94	26.36	29.02	10.06	26.75	38.36	16.24	29.62
LG-f500-d5-n	500	37.99	17.37	35.80	24.39	28.65	9.55	26.45	38.02	16.34	29.80
LG-f1000-d5-n	1,000	37.60	16.52	35.21	26.51	27.94	8.78	25.09	37.34	15.82	29.26
NEG-f100	100	37.64	16.80	35.26	24.50	27.73	8.86	25.24	37.38	15.77	28.84
NEG-f200	200	38.31	17.14	35.94	24.94	28.41	8.63	25.62	37.64	15.76	28.85
NEG-f500	500	38.56	17.56	36.15	24.97	29.27	9.87	26.61	37.89	16.00	29.09
NEG-f1000	1,000	37.74	17.04	35.48	24.69	27.90	8.82	24.95	37.91	15.86	28.89
NEG-LG-f100-d5-n	100	37.24	16.39	34.95	25.72	29.21	9.94	26.66	37.31	15.19	28.90
NEG-LG-f200-d5-n	200	37.33	16.26	34.92	26.75	29.38	9.27	26.46	37.51	14.91	28.84
NEG-LG-f500-d5-n	500	38.36	16.80	35.77	24.14	28.77	9.26	25.83	36.89	14.74	28.43
NEG-LG-f1000-d5-n	1,000	37.60	16.73	35.26	24.49	28.41	9.58	25.57	36.33	14.08	27.98
Baseline	–	36.12	15.36	33.96	28.09	26.72	8.16	24.36	36.88	15.53	29.14

Table 6
ROUGE and NTR scores for: (i) W-LG, W-NEG, and W-NEG-LG strategies, (ii) varying θ_f ($\theta_d = 5$), and (iii) nouns only ($p_{value} < 0.01$).

Model	θ_f	Gigaword				DUC 2004			CNN/DailyMail		
		R-1	**R-2**	**R-L**	**NTR %**	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
W-LG-f100-d5-n	100	37.00	15.21	34.30	18.77	28.35	8.66	25.78	37.69	15.69	29.15
W-LG-f200-d5-n	200	36.27	15.73	33.69	18.91	28.79	8.56	26.36	37.40	15.51	29.05
W-LG-f500-d5-n	500	36.09	15.45	33.90	17.67	27.99	9.42	25.59	37.23	15.27	29.04
W-LG-f1000-d5-n	1,000	35.53	14.23	32.97	18.71	27.28	8.12	24.52	37.19	15.02	28.75
W-NEG-f100	100	35.43	14.34	33.00	19.27	27.18	8.34	24.31	34.50	11.52	26.45
W-NEG-f200	200	34.71	13.89	32.19	19.38	27.75	8.12	24.84	34.39	11.34	26.38
W-NEG-f500	500	34.33	14.00	32.17	19.58	26.77	8.15	24.57	32.17	9.30	20.93
W-NEG-f1000	1,000	33.55	13.89	31.46	19.90	26.11	7.68	23.57	32.90	9.74	20.88
W-NEG-LG-f100-d5-n	100	35.10	14.21	32.63	19.89	27.89	8.15	25.50	34.37	11.40	26.47
W-NEG-LG-f200-d5-n	200	35.14	14.04	32.71	19.22	27.88	8.23	25.43	34.28	11.18	26.50
W-NEG-LG-f500-d5-n	500	34.54	13.63	32.37	19.65	27.15	8.51	24.40	33.75	11.03	26.43
W-NEG-LG-f1000-d5-n	1,000	33.74	13.51	31.66	20.17	26.65	7.85	24.27	33.56	10.79	25.90
WSD-n (Baseline)	–	35.69	14.71	33.39	18.47	27.63	9.15	25.21	36.40	14.67	28.29

the original text, upon which the deep learning model is trained. During evaluation, the WSD-n model generates summaries of a disambiguated version of the input text. Even though no generalization strategy is applied to WSD-n, the WSD identifiers of the generated summary still need to be converted to appropriate words. This is achieved through LG-based post-processing (Section 6) that matches concepts of the original text to the predicted concepts of the generated summary.

In general, WSD-based approaches achieve decreased ROUGE and NTR scores, when compared with non-WSD based ones (Table 5). Additionally, W-NEG and W-NEG-LG exhibit poorer performance when compared to the baseline (WSD-n), with W-LG being an exception, as it performs better.

Table 7 displays the results for both noun and verb generalization for LG and NEG-LG. NEG is not considered in this case, because it is only capable of generalizing nouns. Nevertheless, the performance of the mixed NEG-LG model is presented, because it consists of an LG module that is capable of generalizing both lexical categories. Hyperparameters θ_d, θ_f are set as before (Tables 5 and 6) and the addition of verbs

Table 7
ROUGE and NTR scores for: (i) LG, NEG, and NEG-LG strategies, (ii) varying θ_f ($\theta_d = 5$), and (iii) nouns and verbs ($p_{value} < 0.01$).

Model	θ_f	Gigaword				DUC 2004			CNN/DailyMail		
		R-1	**R-2**	**R-L**	**NTR** %	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
LG-f100-d5-nv	100	39.18	17.41	36.62	24.05	29.17	8.87	26.34	38.66	16.74	30.00
LG-f200-d5-nv	200	39.32	17.71	36.73	25.02	29.07	9.41	26.61	38.56	16.61	29.90
LG-f500-d5-nv	500	38.81	17.32	36.31	24.70	28.46	8.20	26.71	37.91	16.05	29.51
LG-f1000-d5-nv	1,000	37.91	17.59	35.69	25.38	28.46	8.20	25.79	37.89	16.05	29.42
NEG-LG-f100-d5-nv	100	37.56	17.39	35.41	27.86	29.47	9.32	26.41	37.46	15.17	28.59
NEG-LG-f200-d5-nv	200	38.49	16.88	35.89	24.30	29.51	9.85	26.80	36.85	14.52	28.36
NEG-LG-f500-d5-nv	500	38.24	16.99	35.74	24.64	28.74	9.36	26.16	36.47	14.49	28.21
NEG-LG-f1000-d5-nv	1,000	37.43	16.33	34.97	26.14	28.67	9.5	25.94	35.99	13.97	27.85

Table 8
ROUGE and NTR scores for: (i) W-LG, W-NEG, and W-NEG-LG strategies, (ii) varying θ_f ($\theta_d = 5$), and (iii) nouns and verbs ($p_{value} < 0.01$).

Model	θ_f	Gigaword				DUC 2004			CNN/DailyMail		
		R-1	**R-2**	**R-L**	**NTR** %	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
W-LG-f100-d5-nv	100	34.00	12.82	31.94	22.73	26.87	7.14	24.39	34.97	13.13	27.08
W-LG-f200-d5-nv	200	33.43	12.18	31.30	22.52	26.82	7.80	24.41	34.07	12.66	26.55
W-LG-f500-d5-nv	500	33.47	11.93	31.29	23.20	26.38	7.29	24.08	34.06	12.50	26.54
W-LG-f1000-d5-nv	1,000	33.06	12.06	30.90	23.67	26.29	7.51	23.78	33.83	12.60	26.43
W-NEG-LG-f100-d5-nv	100	31.73	11.90	29.71	23.26	25.22	7.37	22.87	31.46	9.36	24.60
W-NEG-LG-f200-d5-nv	200	32.21	11.50	30.14	23.43	25.44	7.65	23.40	31.44	9.20	24.59
W-NEG-LG-f500-d5-nv	500	32.59	11.66	30.41	23.60	25.66	6.64	23.45	30.77	8.88	23.94
W-NEG-LG-f1000-d5-nv	1,000	31.52	11.24	29.55	23.98	24.86	6.44	22.55	30.68	8.66	23.90
WSD-nv (baseline)	–	33.18	12.28	31.02	22.84	25.75	6.93	23.48	33.41	12.01	26.24

as candidate concepts for generalization results in a further improvement, in terms of the ROUGE scores.

Finally, Table 8 reports the performance of W-LG and W-NEG-LG for noun and verb generalization. Similar assumptions to the previous cases regarding hyperparameters, pre-processing, and post-processing are made in this case, as well. WSD-nv serves as the baseline, in a similar fashion to WSD-n being the baseline on Table 6. Even though most of the W-LG models outperform the baseline, overall, they exhibit poor performance, when compared to the models appearing on the other result tables or the state-of-the-art approaches.

8.1 Greedy Concept Matching ROUGE Scores

The results reported above have been obtained using optimal concept matching. Nevertheless, the experiments have been repeated, using greedy concept matching the second time, prior to concluding that they are essentially the same. In particular, in the Gigaword and DUC data sets (TS of short documents) differences are reported in the third decimal digit, while in the CNN/DailyMail data set (document level TS), differences are reported in the second decimal digit. Consequently, we conclude that in short TS, the optimal and greedy concept matching produce the same summaries and therefore we do not report separate results for these two matching methods. In the case of the CNN/Dailymail data set, to indicate the degree of differentiation between the two methods, in Table 9, we report the ROUGE scores of LG and NEG strategies

Table 9
ROUGE scores of using greedy concept matching and the differences from the optimal matching in CNN/DailyMail data set for: (i) LG and NEG strategies, (ii) varying θ_f ($\theta_d = 5$), and (iii) nouns ($p_{value} < 0.01$).

Model	**ROUGE-1**		**ROUGE-2**		**ROUGE-L**	
LG-f100-d5-n	38.77	0.00	16.78	−0.01	29.87	0.00
LG-f200-d5-n	38.33	−0.03	16.24	0.00	29.60	−0.02
LG-f500-d5-n	37.93	−0.09	16.16	−0.18	29.69	−0.11
LG-f1000-d5-n	37.34	0.00	15.68	−0.14	29.21	−0.05
NEG-f100	37.38	0.00	15.67	−0.10	28.78	−0.06
NEG-f200	37.63	−0.01	15.71	−0.05	28.77	−0.08
NEG-f500	37.83	−0.06	15.91	−0.09	29.03	−0.06
NEG-f1000	37.84	−0.07	15.84	−0.02	28.81	−0.08

for greedy concept matching. Comparing them with the optimal one (Table 5), next to each ROUGE score of Table 9, we provide its difference from the respective score of the optimal solution.

8.2 Results on the PG, RL, TR, and PETR models

Table 10 illustrates the performance of the PG, RL, TR, and PETR models on the ROUGE metrics for the LG strategy, for varying thresholds of minimum word frequency for generalization ($\theta_f = \{100, 200, 500, 1{,}000\}$). The last four rows report the performance of the baseline models, where no generalization strategy has been applied (Section 7.4). It is evident that our proposed framework outperforms the baselines and the respective models of the attentive seq2seq network (Table 5). Finally, the combination of the LG strategy with the PG, RL, TR, or PETR model achieves promising results, comparable to the most recent competitive approaches of Tables 3 and 4, for the respective data sets.

8.3 Factual Accuracy Results

Because factual consistency is related to the field of automatic TS (Kryściński et al. 2019; Zhang 2019; Goodrich et al. 2019), Table 11 summarizes the factual accuracy (Section 7.2) of the *LG, NEG, NEG-LG, W-LG, W-NEG*, and *W-NEG-LG* generalization strategies. The reported values represent the average factual accuracy of the instances of the test set for each data set, for the attentive seq2seq model. Additionally, the same table outlines the performance of the baseline (Section 7.4) and *WSD-n* models, which (the latter) also constitutes a baseline for the WSD-based models, as it has also been mentioned above. Moreover, the last row of this table reports the factual accuracy of the reference summary, where, in the case of the DUC data set, the value corresponds to the average factual accuracy of the four reference summaries that are provided for each instance.

In the same manner, Table 12 reports factual accuracy assuming noun and verb generalization. The *NEG* models are also absent from this table because the *NEG* strategy generalizes only noun entities, while here both noun and verb generalization is required. Moreover, the table outlines the performance of the baseline (Section 7.4) and *WSD-nv* models, which (the latter) also constitutes a baseline for the WSD-based models.

Table 10
ROUGE scores of pointer-generator (PG), reinforcement learning (RL), transformer (TR), and pretrained encoder transformer (PETR) networks for: (i) LG strategy, (ii) varying θ_f ($\theta_d = 5$), and (iii) noun generalization ($p_{value} < 0.01$).

Model	Gigaword			DUC 2004			CNN/DailyMail		
	ROUGE-1	**ROUGE-2**	**ROUGE-L**	**ROUGE-1**	**R-2**	**ROUGE-L**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
LG-f100-d5-n (PG)	40.94	19.65	38.52	29.66	10.49	27.09	43.04	20.65	30.93
LG-f200-d5-n (PG)	40.26	19.25	37.99	29.47	09.78	26.59	42.15	20.19	30.23
LG-f500-d5-n (PG)	40.23	19.42	37.90	28.83	09.53	25.94	41.89	19.30	30.23
LG-f1000-d5-n (PG)	39.85	19.02	37.78	28.51	10.34	25.86	41.18	18.79	29.22
LG-f100-d5-n (RL)	41.72	20.57	39.31	29.97	10.35	27.17	43.63	20.87	31.46
LG-f200-d5-n (RL)	41.57	20.37	39.18	29.84	10.36	26.45	43.33	20.60	31.45
LG-f500-d5-n (RL)	41.11	20.05	38.83	28.92	10.23	26.20	42.36	20.01	30.48
LG-f1000-d5-n (RL)	40.67	19.64	38.44	28.80	10.18	26.09	41.60	19.40	29.87
LG-f100-d5-n (TR)	41.35	20.09	38.89	29.87	10.45	27.11	43.33	20.78	34.16
LG-f200-d5-n (TR)	40.91	19.84	38.49	29.59	10.08	26.50	42.70	20.38	33.81
LG-f500-d5-n (TR)	40.71	19.70	38.36	28.91	10.02	26.14	42.16	19.65	33.44
LG-f1000-d5-n (TR)	40.28	19.43	38.10	28.59	10.27	26.01	41.38	19.42	32.57
LG-f100-d5-n (PETR)	42.12	20.76	39.50	29.91	10.34	27.32	43.93	21.07	35.96
LG-f200-d5-n (PETR)	41.86	20.42	39.21	29.93	10.31	26.55	43.64	20.58	34.72
LG-f500-d5-n (PETR)	41.32	20.25	38.77	29.12	10.20	26.39	42.52	20.31	33.75
LG-f1000-d5-n (PETR)	40.59	19.68	38.12	28.60	10.09	26.14	41.72	19.54	33.71
Baseline (PG)	39.33	18.31	37.10	27.45	09.72	25.60	41.40	19.26	29.56
Baseline (RL)	40.57	19.72	38.36	27.82	09.79	25.75	41.67	19.34	29.42
Baseline (TR)	40.18	18.96	37.26	27.43	09.75	25.65	41.08	18.12	33.37
Baseline (PETR)	40.95	19.78	38.61	27.91	09.85	26.05	41.79	19.38	35.62

Table 11
Factual accuracy for: (i) LG, NEG, NEG-LG, W-LG, W-NEG, and W-NEG-LG strategies, (ii) varying $\theta_f = \{100, 200, 500, 1{,}000\}$, $\theta_d = 5$, and (iii) nouns.

Model	Gigaword	DUC	CNN/DM	Model	Gigaword	DUC	CNN/DM
		fact$_{acc}$ %				*fact*$_{acc}$ %	
LG-f100-d5-n	47.21	41.31	69.48	W-LG-f100-d5-n	49.82	38.48	68.55
LG-f200-d5-n	43.14	36.01	67.76	W-LG-f200-d5-n	45.35	39.61	64.94
LG-f500-d5-n	45.28	39.40	67.98	W-LG-f500-d5-n	45.83	45.07	66.95
LG-f1000-d5-n	42.20	40.30	62.24	W-LG-f1000-d5-n	44.87	42.68	62.87
NEG-f100-n	45.74	41.42	65.34	W-NEG-100-d5-n	47.18	36.64	35.55
NEG-f200-n	46.39	37.60	67.12	W-NEG-f200-n	43.24	38.31	37.28
NEG-f500-n	43.84	39.16	66.85	W-NEG-f500-n	42.94	41.25	54.02
NEG-f1000-n	44.96	40.56	70.70	W-NEG-f1000-n	43.02	37.05	54.42
NEG-LG-f100-d5-n	45.56	37.13	61.74	W-NEG-LG-f100-d5-n	45.93	39.70	37.22
NEG-LG-f200-d5-n	45.66	40.83	61.73	W-NEG-LG-f200-d5-n	45.15	37.03	27.84
NEG-LG-f500-d5-n	45.83	37.20	63.80	W-NEG-LG-f500-d5-n	45.32	35.79	25.47
NEG-LG-f1000-d5-n	46.42	41.83	59.36	W-NEG-LG-f1000-d5-n	46.95	43.61	34.17
Baseline	41.98	39.55	54.47	WSD-n (baseline)	45.70	48.27	60.75
Reference	32.71	17.36	32.56	–	–	–	–

8.4 Case Study

To further illustrate the workflow and the main aspects of the framework for generating the final summaries, Tables 13 and 14 present examples of generated summaries using the LG, NEG, and NEG-LG strategies for noun generalization. In particular, in the case of LG strategy, during the pre-processing task (Section 4), some words that have an insufficient number of usage examples in the data set have been generalized, using the *WordNet* (Miller 1995; Fellbaum 1998) sense identifiers (e.g., *produce.n.01*). The utilized

Table 12
Factual accuracy for: (i) LG, NEG-LG, W-LG, and W-NEG-LG strategies, (ii) varying $\theta_f = \{100, 200, 500, 1{,}000\}$, $\theta_d = 5$, and (iii) nouns and verbs.

Model	Gigaword	DUC	CNN/DM	Model	Gigaword	DUC	CNN/DM
		$fact_{acc}$ %				$fact_{acc}$ %	
LG-f100-d5-nv	46.86	41.92	70.42	W-LG-f100-d5-nv	43.76	32.29	24.81
LG-f200-d5-nv	44.85	35.20	68.18	W-LG-f200-d5-nv	42.82	33.00	26.87
LG-f500-d5-nv	43.89	40.04	65.29	W-LG-f500-d5-nv	45.82	39.08	23.97
LG-f1000-d5-nv	43.06	39.26	67.68	W-LG-f1000-d5-nv	46.54	38.16	22.39
NEG-LG-f100-d5-nv	43.23	37.80	63.72	W-NEG-LG-f100-d5-nv	42.59	38.60	9.05
NEG-LG-f200-d5-nv	46.11	38.80	59.03	W-NEG-LG-f200-d5-nv	45.25	35.85	8.09
NEG-LG-f500-d5-nv	45.82	39.08	58.14	W-NEG-LG-f500-d5-nv	43.49	35.64	6.61
NEG-LG-f1000-d5-nv	45.39	37.98	57.02	W-NEG-LG-f1000-d5-nv	43.04	37.31	8.14
–	–	–	–	WSD-nv (baseline)	45.90	40.38	17.82

machine learning model (here the attentive seq2seq architecture of Section 5.2.1 has been used) is trained to predict the system summary. Then, the post-processing task (Section 6) generates the final summary. In the example of short TS in Table 13, we can see that the system summary based on the LG strategy contains the generalized sense *produce.n.01* (*"fresh fruits and vegetable grown for the market"*, according to *WordNet* sense). In post-processing, this generalized token is replaced by the word *tomatoes* from the input text, which is a hyponym of the sense *produce.n.01*, forming the final summary. In this example of the LG strategy, all tokens of the final summary are included in the input text, and also, the phrases *"to close out"* and *"of summer"* exist in the input text. Similarly, in the example of document-level TS (Table 14), in the generalized text of the LG strategy, we have the generalized sense *copycat.n.01* that also appears in the predicted summary. In the final summary, this sense has been replaced by the word *parrot*. Also, the final summary of this example contains phrases of the original text and new words such as *oregon* that do not appear in the original text.

Table 13
Examples of short TS from the input text to the output summary for the LG, NEG, and NEG-LG strategies of noun generalization.

	Input text: plump, juicy, bright red tomatoes that hang heavy on the vine; basil that grows profusely, and so many cucumbers that you run out of ideas for using them: these are just the foods to close out the hot days of summer, and to provide a final note for this column, which will cease this week .
LG	**Generalized Text:** plump, juicy, bright red produce.n.01 that hang heavy on the plant.n.02; flavorer.n.01 that grows profusely, and so many produce.n.01 that you run out of ideas for using them: these are just the foods to close out the hot days of summer, and to provide a final note for this column, which will cease this week . **System Summary:** produce.n.01 to close out of summer **Final Summary** tomatoes to close out of summer
NEG	**Generalized Text:** plump, juicy, bright red tomatoes that hang heavy on the vine; basil that grows profusely, and so many cucumbers that you run out of ideas for using them: these are just the foods to close out DATE, and to provide a final note for this column, which will cease this week . **System Summary:** vegetables to close this DATE **Final Summary** vegetables to close this summer
NEG-LG	**Generalized Text:** plump, juicy, bright red produce.n.01 that hang heavy on the plant.n.02; flavorer.n.01 that grows profusely, and so many produce.n.01 that you run out of ideas for using them: these are just the foods to close out DATE, and to provide a final note for this column, which will cease this week . **System Summary:** vegetable.n.01 to close this DATE **Final Summary:** foods to close this summer
	Reference summary: hail to summer 's end and farewell

Table 14
Examples of document-level TS from the input text to the output summary for the LG, NEG, and NEG-LG strategies of noun generalization.

	Input Text: firefighters responded to cries for help - from two parrots . the crew scoured a burning home in boise, idaho, searching for people shouting ′ help ! ′ and ′ fire ! ′ eventually, to their surprise, they found a pair of squawking birds . scroll down for video . cry for help ! this is one of the two parrots who were found in a burning home after calling for help . the tropical creatures appeared to have been alone when flames began to sweep the property . but they seemed to know what to do . both were pulled from the home and given oxygen . they are expected to survive . the fire crew in boise, idaho, thought they were chasing human voices when the found the birds . treatment: the officials treated the birds with oxygen masks and survive . according to kboi, the cause of the officers managed to contain the fire to just one room . it is being both are expected to investigated and no people were found inside . officials have yet to track down the birds ′ owners .
LG	**Generalized Text:** firefighters responded to cries for help - from two copycat.n.01 . the crew scoured a burning home in boise, idaho, searching for people shouting ′ help ! ′ and ′ fire ! ′ eventually, to their astonishment.n.01, they found a pair of squawking birds . scroll down for video . cry for help ! this is one of the two copycat.n.01 who were found in a burning home after calling for help . [...] yet to track down the birds ′ owners . **System Summary:** the two copycat.n.01 were found in a burning home in oregon, idaho . they were pulled from the home and given oxygen . they are expected to survive . **Final Summary:** the two parrot were found in a burning home in oregon, idaho . they were pulled from the home and given oxygen . they are expected to survive .
NEG	**Generalized Text:** firefighters responded to [...] a burning home in GPE, GPE, searching for people shouting [...] according to ORG, the cause of the officers managed to contain [...] the birds ′ owners . **System Summary:** the fire crew in GPE, GPE, thought they were chasing human voices . they were found in a burning home in GPE, GPE . it is being investigated and no people were found inside . **Final Summary:** the fire crew in boise, idaho, thought they were chasing human voices . they were found in a burning home in boise, idaho . it is being investigated and no people were found inside .
NEG-LG	**Generalized Text:** firefighters responded to cries for help - from two copycat.n.01 . the crew scoured a burning home in GPE, GPE, searching for people shouting ′ help ! ′ and ′ fire ! ′ eventually, to their astonishment.n.01, they found a pair of squawking birds . scroll down for video . cry for help ! this is one of the two copycat.n.01 who were found in a burning home [...]. the fire crew in GPE, GPE, thought they were chasing human voices [...] to survive . according to ORG, the cause of the officers managed to [...] to track down the birds ′ owners . **System Summary:** the crew scoured a burning home in GPE, GPE, searching for people shouting ′ help ! ′ and ′ fire ! ′ they found the copycat.n.01 with oxygen masks and both are expected to survive . **Final Summary:** the crew scoured a burning home in boise, idaho, searching for people shouting ′ help ! ′ and ′ fire ! ′ they found the parrots with oxygen masks and both are expected to survive .
	Reference summary: two parrots were home alone when a fire erupted in boise, idaho . started calling ′ help ! ′ and ′ fire ! ′ , crew thought they were human voices . both were pulled from the wreckage and treated with oxygen masks .

In the example of the NEG strategy (Table 13), the generalized named entity *DATE*, which appears in the predicted summary, has been replaced by the word *summer* (from the input text) in the post-processing phase that generates the final summary. Moreover, in the same example, the word *vegetables*, which appears both in the predicted and final summaries, is a new word that does not appear in the input text. The appearance of new words in the final summaries indicates that the system not only copies content from the input text but also is capable of generating new words or phrases, following the meaning of the input text. The sense *vegetable.n.01* also appears in the system summary of the NEG-LG strategy, where, in post-processing, this sense has been replaced by the word *food* because the algorithm tries to select the most relevant word from the input text. Similarly, in the example of the NEG strategy of document level summarization (Table 14), the named entities in the system summary have been replaced by the appropriate words of the original text, in order to generate the final summary. Finally, in the case of the NEG-LG strategy, it becomes apparent that we have a combination of both the NEG and LG methods, as illustrated in the present examples.

9. Discussion

Overall, the experimental procedure aims at examining various aspects of the proposed methodology. Initially, the effect of taxonomy depth in generalization is discussed (Section 9.1), followed by that of concept frequency (Section 9.2). Then, the influence of WSD (Section 4.1) and POS (Section 9.4) are also assessed. Finally, Section 9.5 overviews the obtained results on the NTR metric, Section 9.7 discusses factual consistency of generated summaries, Section 9.8 overviews the perspective of enhancing seq2seq models, while in Section 9.9, some concluding remarks concerning the framework as a whole, are made.

9.1 The Effect of the Taxonomy Depth

In principle, LG-based strategies are able to generalize concepts ranging from very general (low taxonomy depth) to more specific (high taxonomy depth). Figure 6 displays the ROUGE scores for varying levels of generalization ($\theta_d \in \{3, 4, 5, 6, 7\}$). The optimum taxonomy depth is 5; when $\theta_d < 5$, the performance decreases as these levels of generalization lead to very general concepts. For $\theta_d > 5$, the ROUGE scores fall again, because the grater taxonomy depth restricts generalization, as fewer concepts can be generalized.

9.2 The Effect of Concept Frequency

As has already been discussed in Section 4.2, the proposed frameworks deals with the problem of OOV or rare words by imposing a minimum frequency (θ_f) below which they need to be generalized. In the experiments, four different frequency thresholds have been considered ($\theta_f \in \{100, 200, 500, 1{,}000\}$) for all generalization strategies.

LG and W-LG achieve their best performance for lower thresholds (100 and 200, respectively) on data sets. For instance, when only noun generalization is considered, LG-f100-d5-n and W-LG-f100-d5-n (Tables 5 and 6) maximize their efficiency for $\theta_f = 100$ in terms of ROUGE-1 or for $\theta_f = 200$ in terms of ROUGE-L. This is also the case of LG-f100-d5-nv and W-LG-f100-d5-nv (Tables 7 and 8); that is, when both nouns and verbs are taken into account. However, in LG, the generalization of rare words maximizes performance, while the generalization of more frequent concepts (e.g., when $\theta_f = 500$ or $\theta_f = 1{,}000$) reduces the ROUGE scores.

In the case of the NEG strategy (Table 5), the models tend to exhibit high performance even if they generalize frequent concepts. In particular, NEG-f500 ($\theta_f = 500$) achieves the highest ROUGE scores on three data sets. This is attributed to the fact that named entities have a similar function in language. Additionally, the post-processing task is capable of matching a named entity identifier to its particular concept, even if concepts of high frequency have been generalized. On the other, W-NEG models (Table 6) achieve their best results for $\theta_f = 100$, as they are based on the disambiguated text, which increases the vocabulary size. NEG-LG and W-NEG-LG obtain their best output mainly for $\theta_f = 200$ or $\theta_f = 500$, because they actually combine both strategies (NEG and LG). Those thresholds may be viewed as a trade-off between NEG, which boosts generalization of frequent words and LG, which is more efficient when rare concepts are generalized.

It is obvious that the choice of an appropriate frequency threshold (θ_f) greatly affects the accuracy of the produced summaries. The obtained results reveal that LG works better when rare words are generalized, NEG prefers the generalization of more frequent words, and NEG-LG seeks a trade-off between the two. Additionally, all approaches exhibit poorer performance when very frequent words ($\theta_f = 1{,}000$) are generalized. In the latter case, the high threshold value results in over-generalization, grouping several words to the same sense. Therefore, the reduced number of synonyms (or words of close meaning) restricts the capability of predicting an appropriate term for a particular context and makes it difficult for the post-processing task to match the general entities to specific concepts.

9.3 The Effect of WSD

WSD is used to identify the different concepts in the text, generalizing them to an appropriate sense. Table 5 presents the results for noun-only generalization, while in Table 7 both noun and verb generalization is considered, with the respective systems outperforming the state-of-the-art (Table 3), with respect to ROUGE scores. The positive experimental results, especially in LG and NEG strategies, are due to the fact that WSD aims at performing an accurate content generalization in the pre-processing phase. Additionally, in the post-processing phase, the WSD identifiers are used for achieving an efficient concept matching (i.e., replacing the general concepts with specific ones by giving priority to particular concepts, or weighting the concepts that belong to the same hyponym or hypernym semantic paths with the general ones).

Even though WSD is efficient for text generalization, the conversion of the original text to a disambiguated version fails to bring about any further improvement. This is evident in Tables 6 and 8, where the WSD-based models exhibit worse performance, compared with the non-WSD-based ones (Tables 5 and 7). The WSD-based models fail to attain satisfactory performance because the universally disambiguated text increases the vocabulary of the data set while simultaneously decreasing the number of occurrences of each sense in the text, thereby affecting the ability of a machine learning model to be trained sufficiently. Moreover the post-processing phase is not capable of replacing a vast amount of senses, represented by WSD identifiers, with the appropriate words of the original text. Nevertheless, comparing WSD-based noun generalization (Table 6) and its noun and verb counterpart (Table 8) with their baselines (WSD-n and WSD-nv, respectively), it becomes evident that the utilized generalization strategies result in increased ROUGE scores. This is attributed to the fact that the generalization methodology reduces the vocabulary size, thereby increasing word frequency in text. This constitutes an indication that the proposed framework improves the performance of text summarization, even though the utilized data set is a disambiguated version of the original corpus.

It should be noted that in the case of the `Gigaword` data set, the numerical digits have been replaced by the number sign (#) (Rush, Chopra, and Weston 2015). Because the knowledge-based WSD method proposed in our work does not identify numbers, the said replacement does not affect system performance. Unlike WSD, NER is capable of recognizing tokens of numerical digits of the respective named entities (e.g., DATE, TIME, PERCENT, MONEY, QUANTITY).

9.4 The Effect of POS

LG-based strategies can generalize nouns, verbs, or both. Verb-only generalization is absent from the experiments because of the low frequency of this lexical category in text (Table 1). In contrast, noun generalization brings a considerable performance improvement because of the high frequency of this lexical category in text. When both are considered, system efficiency is slightly enhanced in terms of the ROUGE scores (Tables 5 and 7). The same conclusion cannot be drawn for WSD-based models (Tables 6 and 8), as they fail to exhibit satisfactory performance. In conclusion, LG models that generalize both low frequency nouns and verbs are those with the highest ROUGE scores.

9.5 NTR in Generated Summaries

NTR quantifies the level of abstraction the proposed frameworks achieves, as it captures the percentage of tokens in the generated summary that do not appear in input text. The obtained NTR results of our models are in line with the *Words-lvt2k-1sent* and *Model #8* state-of-the-art approaches (Table 3). More specifically, on Table 5, where the baseline approach exhibits the best performance in terms of NTR, new tokens constitute one in every four words in a summary. In WSD-based models, NTR drops to one new word for every five words in the summary, because of the specific meaning of each token in data, which prohibits the deep learning model from learning the particular function of each term. Generally, models that have several words with similar meaning (i.e., synonyms), like the baseline model, witness increased NTR. In contrast, models with few synonyms (e.g., WSD-based) have a reduced NTR. NTR levels may also be related to the extent of alternative options a model has to generate a word. In particular, if a context can be described by many words or there exist a lot of synonyms in text (i.e., in training set), then higher values of NTR may be observed. On the other hand, a limited number of synonyms result in a reduction of new tokens in a generated summary.

9.6 Greedy and Optimal Concept Matching

The slight differences among the greedy and optimal concept matching (Table 9) are due to the fact that these approaches are aided by the computation of text similarity (Algorithms 4 and 5). In particular, these algorithms weigh the candidate concepts according to their ontology semantic paths (for LG-based strategies) or the specified named entity (for the NEG-based strategies). Therefore, the methodology of concept matching tries to resolve conflicts between two or more concepts that are candidates for replacing the same generalized concept. Because these conflicts appear more often in larger texts (e.g., in CNN/DailyMail), the differences between the greedy and optimal concept matching become more apparent there, but still, they remain insignificant as the experimental results show (Table 9). Finally, greedy concept matching is an efficient and simple method that could be used as an alternative to the optimal one, especially in those cases where an efficient solver for integer linear programming is not provided, or a huge amount of documents needs to be summarized.

9.7 Factual Consistency

To examine the factual consistency of the generated summaries, their factual accuracy is measured, as described in Section 7.2. According to the obtained results (Tables 11 and 12) presented in Section 8.3, the generated summaries for the CNN/DailyMail data set exhibit greater factual accuracy than those of the other two data sets (`Gigaword` and DUC). This is attributed to the fact that in document-level TS, the produced summaries cover more facts of the original text, while in short-document TS, the short summaries are not capable of reflecting the facts of the original text sufficiently. This is particularly evident in the obtained results for the *LG*, *NEG*, and *NEG-LG* strategies on document level TS. Another observation is that, while the summaries of the CNN/DailyMail data set contain more concepts than those of the other data sets (i.e., concepts for performing concept matching in the post-processing phase), the factual accuracy in document level TS is not decreased. Consequently, the factual consistency of the produced summaries is not affected by increasing the number of concepts for concept matching or by increasing the summary length. On the contrary, the longer summaries (i.e., in CNN/DailyMail data set) seem to achieve higher factual accuracy than the shorter ones (e.g., in `Gigaword`).

In a similar manner to ROUGE scores, the factual accuracy of the *LG* strategy reaches its maximum for $\theta_f = 100$ (i.e., generalizing rare concepts), while the *NEG* strategy tends to achieve higher scores for generalizing frequent concepts (e.g., the NEG-f1000-n model exhibiting the highest score in CNN/DailyMail data set). Comparing the factual accuracy of noun generalization (Table 11) with that of both noun and verb generalization (Table 12), we conclude that the addition of verbs in the content generalization phase creates slight differences in *LG, NEG*, and *NEG-LG* models, without any considerable improvement. On the other hand, noun and verb generalization in WSD-based models (*W-LG, W-NEG*, and *W-NEG-LG*) affect negatively the factual consistency. In most cases, WSD-based models fail to produce factual consistent summaries for the same reasons that they fail to achieve satisfactory ROUGE scores mentioned above.

Moreover, the factual accuracy of our framework is improved when compared with that of the baseline models and the reference summaries. This does not necessarily imply that the generated summaries are better than the human-written ones, but rather highlight the fact that authors are able to produce summaries with rephrased content. In machine-produced summaries on the other hand, the system tends to copy facts from the source text (e.g., $NTR = 24.97\%$ in LG-f100-d5-n model of Table 5), allowing the factual accuracy (which measures the coverage of facts between a summary and the respective source text) to achieve high scores. Therefore, the lower score of the reference summaries is not an indication that they are poorly written. On the contrary, the high scores, especially in the case of document level TS (CNN/DailyMail data set), constitute a strong indication that our framework produces factually consistent summaries.

9.8 Enhancing Sequence-to-Sequence Models

The main purpose of this work is to provide a framework capable of enhancing the performance of seq2seq deep learning models in abstractive TS. In this direction, the proposed framework deals with two fundamental issues in machine learning approaches; (i) providing a sufficient number of usage examples in the training phase for achieving accurate predictions, and (ii) adapting new, unseen instances to the specifics of the model. The extensive experimental procedure (Section 7), based on the attentive seq2seq model (Section 5.2.1), demonstrates that the proposed framework is capable of improving performance. Additionally, the use of four additional seq2seq deep learning architectures, the pointer-generator network (Section 5.2.2), the reinforcement learning model (Section 5.2.3), the transformer, and the pretrained encoder transformer architectures (Section 5.2.4) further bolster the versatility of the proposed method.

9.9 Summarizing the Results

The analysis presented so far indicates that the presented framework is an efficient solution for abstractive TS, outperforming baselines and other state-of-the-art systems, especially in the case of the advanced models such as reinforcement learning or transformer-based approaches (Table 10). It has been demonstrated that the level of generalization affects the obtained results and that setting the correct threshold for concept frequency (to be considered for generalization) improves system performance.

WSD-based models' suboptimal performance is attributed to the following reasons: (i) the deep learning system fails to predict the appropriate words, because of the large number of distinct tokens (i.e. tokens with specific meaning), and (ii) the post-processing task fails to match the WSD identifiers to appropriate words, because of the large number of required matches in the predicted summary. Moreover, models based on noun-only generalization exhibit an increased performance, with further improvement, in terms of ROUGE scores, being possible when both verbs and nouns are considered. Furthermore, NTR is reduced when synonyms in text are limited (e.g., in WSD-based models), while this metric is maximized for baseline models that have not applied any generalization strategy that might reduce the number of words with similar meaning. Finally, in assessing the factual consistency, the document level TS achieves higher factual accuracy than that of short document TS.

10. Conclusion and Future Work

In this work, a novel framework for abstractive TS that combines deep learning techniques with knowledge-based methodologies has been proposed. The framework is based on a well-defined theoretical model for generating abstractive summaries. In particular, its components include knowledge resources in ontological representations, WSD, NER, content generalization, word embeddings, deep learning predictions, text similarity, and concept matching. Overall, the methodology aims at improving the performance of a sequence-to-sequence deep learning model.

The performance of the proposed approach has been thoroughly evaluated by an extensive experimental procedure that examined various aspects of the framework on popular data sets (`Gigaword`, `DUC 2004`, and `CNN/DailyMail`). The obtained results have been promising, as they were better than those of the baseline and of other state-of-the-art systems. This is attributed to the appropriate data transformations, model optimization, as well as the ability of the system to deal with OOV or rare words.

Even though this framework already exhibits a satisfactory performance, it could be further enhanced, especially with regard to hyperparameter estimation. In-depth knowledge of those aspects that affect the performance of the framework, as well as a theoretical model for estimating the optimal values of hyperparameters, are expected to yield even better results.

Acknowledgments
This work was supported by computational time granted from the National Infrastructures for Research and Technology S.A. (GRNET S.A.) in the National HPC facility - ARIS - under project ID pa181003-ISLAB-Med.

References

Allahyari, Mehdi, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text summarization techniques: A brief survey. *arXiv preprint arXiv:1707.02268*. `https://doi.org/10.14569/IJACSA.2017.081052`

Alshaina, S., Ansamma John, and Aneesh G. Nath. 2017. Multi-document abstractive summarization based on predicate argument structure. In *Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2017 IEEE International Conference on*, pages 1–6. `https://doi.org/10.1109/SPICES.2017.8091339`

Andor, Daniel, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and

Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*. https://doi.org/10.18653/v1/P16-1231

Angeli, Gabor, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354. https://doi.org/10.3115/v1/P15-1034

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Banerjee, Satanjeev and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using WordNet. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 136–145. Springer. https://doi.org/10.1007/3-540-45715-1_11

Banerjee, Satanjeev and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *IJCAI*, volume 3, pages 805–810.

Baralis, Elena, Luca Cagliero, Saima Jabeen, Alessandro Fiori, and Sajid Shah. 2013. Multi-document summarization based on the Yago ontology. *Expert Systems with Applications*, 40(17):6976–6984. https://doi.org/10.1016/j.eswa.2013.06.047

Barzilay, Regina and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328. https://doi.org/10.1162/089120105774321091

Bertsekas, Dimitri P. 1998. *Network Optimization: Continuous and Discrete Models*. Belmont: Athena Scientific.

Bizer, Christian, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia - a crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165. https://doi.org/10.1016/j.websem.2009.07.002

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146. https://doi.org/10.1162/tacl_a_00051

Borah, Pranjal Protim, Gitimoni Talukdar, and Arup Baruah. 2014. Approaches for word sense disambiguation—A survey. *International Journal of Recent Technology and Engineering*, 3(1):35–38.

Boulanger-Lewandowski, Nicolas, Yoshua Bengio, and Pascal Vincent. 2013. Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340, Citeseer.

Brunsch, Tobias, Kamiel Cornelissen, Bodo Manthey, and Heiko Röglin. 2013. Smoothed analysis of belief propagation for minimum-cost flow and matching. In *International Workshop on Algorithms and Computation*, pages 182–193, Springer. https://doi.org/10.1007/978-3-642-36065-7_18

Celikyilmaz, Asli, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. *arXiv preprint arXiv:1803.10357*. https://doi.org/10.18653/v1/N18-1150

Chaplot, Devendra Singh and Ruslan Salakhutdinov. 2018. Knowledge-based word sense disambiguation using topic models. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chen, Yen-Chun and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*.

Choi, Jinho D., Joel Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 387–396. https://doi.org/10.3115/v1/P15-1038

Chopra, Sumit, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98. https://doi.org/10.18653/v1/N16-1012

Cohan, Arman, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. *arXiv preprint arXiv:1804.05685*. https://doi.org/10.18653/v1/N18-2097

Cohn, Trevor and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 137–144. https://doi.org/10.3115/1599081.1599099

Dasgupta, Sanjoy, Christos H. Papadimitriou, and Umesh V. Vazirani. 2008. *Algorithms*. McGraw-Hill.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Edmundson, Harold P. 1969. New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285. https://doi.org/10.1145/321510.321519

Fellbaum, Christiane. 1998. *WordNet: An Electronic Lexical Database*. MIT Press. https://doi.org/10.7551/mitpress/7287.001.0001

Filippova, Katja. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 322–330.

Filippova, Katja and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 177–185. https://doi.org/10.3115/1613715.1613741

Gambhir, Mahak and Vishal Gupta. 2017. Recent automatic text summarization techniques: A survey. *Artificial Intelligence Review*, 47(1):1–66. https://doi.org/10.1007/s10462-016-9475-9

Gao, Yang, Yang Wang, Luyang Liu, Yidi Guo, and Heyan Huang. 2020. Neural abstractive summarization fusing by global generative topics. *Neural Computing and Applications*, 32(9):5049–5058.

Gehrmann, Sebastian, Yuntian Deng, and Alexander M. Rush. 2018. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*. https://doi.org/10.18653/v1/D18-1443

Genest, Pierre Etienne and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 64–73.

Golik, Pavel, Patrick Doetsch, and Hermann Ney. 2013. Cross-entropy vs. squared error training: A theoretical and experimental comparison. In *Interspeech*, 13:1756–1760. https://doi.org/10.21437/Interspeech.2013-436

Goodrich, Ben, Vinay Rao, Peter J. Liu, and Mohammad Saleh. 2019. Assessing the factual accuracy of generated text. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 166–175. https://doi.org/10.1145/3292500.3330955

Graves, Alex. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*. https://doi.org/10.1007/978-3-642-24797-2_3

Graves, Alex, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. https://doi.org/10.1109/ASRU.2013.6707742

Gupta, Som and S. K. Gupta. 2019. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121:49–65. https://doi.org/10.1016/j.eswa.2018.12.011

Hansen, Ben B. and Stephanie Olsen Klopfer. 2006. Optimal full matching and related designs via network flows. *Journal of Computational and Graphical Statistics*, 15(3):609–627. https://doi.org/10.1198/106186006X137047

Hennig, Leonhard, Winfried Umbrath, and Robert Wetzker. 2008. An ontology-based approach to text summarization. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 291–294. https://doi.org/10.1109/WIIAT.2008.175

Hermann, Karl Moritz, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Hípola, Pedro, José A. Senso, Amed Leiva-Mederos, and Sandor Domínguez-Velasco. 2014. Ontology-based text summarization. The case of texminer. *Library Hi Tech*, 32(2):229–248. https://doi.org/10.1108/LHT-01-2014-0005

Honnibal, Matthew and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language*

Processing, pages 1373–1378. https://doi.org/10.18653/v1/D15-1162

Joshi, Akanksha, E. F. Fernández, and E. Alegre. 2018. Deep learning based text summarization: Approaches databases and evaluation measures. In *International Conference of Applications of Intelligent Systems*.

Joshi, Monika, Hui Wang, and Sally McClean. 2018. Dense semantic graph and its application in single document summarisation. In *Emerging Ideas on Information Filtering and Retrieval*. Springer, pages 55–67. https://doi.org/10.1007/978-3-319-68392-8_4

Keneshloo, Y., T. Shi, N. Ramakrishnan, and C. K. Reddy. 2020. Deep reinforcement learning for sequence-to-sequence models. *IEEE Transactions on Neural Networks and Learning Systems*, 31(7):2469–2489.

Khan, Atif, Naomie Salim, Haleem Farman, Murad Khan, Bilal Jan, Awais Ahmad, Imran Ahmed, and Anand Paul. 2018. Abstractive text summarization based on improved semantic graph approach. *International Journal of Parallel Programming*, pages 1–25. https://doi.org/10.1007/s10766-018-0560-3

Kingma, Diederik P. and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Knight, Kevin and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107. https://doi.org/10.1016/S0004-3702(02)00222-9

Kouris, Panagiotis, Georgios Alexandridis, and Andreas Stafylopatis. 2019. Abstractive text summarization based on deep learning and semantic content generalization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5082–5092, Florence. https://doi.org/10.18653/v1/P19-1501

Kovács, Péter. 2015. Minimum-cost flow algorithms: An experimental evaluation. *Optimization Methods and Software*, 30(1):94–127. https://doi.org/10.1080/10556788.2014.895828

Kryściński, Wojciech, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Evaluating the factual consistency of abstractive text summarization. *arXiv preprint arXiv:1910.12840*.

Kusner, Matt, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Lee, Chang Shing, Zhi-Wei Jian, and Lin-Kai Huang. 2005. A fuzzy ontology and its application to news summarization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(5):859–880. https://doi.org/10.1109/TSMCB.2005.845032, PubMed: 16240764

Li, Yang and Tao Yang. 2018. *Word Embedding for Understanding Natural Language: A Survey*. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-53817-4_4

Li, Yuxi. 2018. Deep reinforcement learning. *arXiv preprint arXiv:1810.06339*.

Lin, Chin-Yew. 2004. ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Lin, Hui and Vincent Ng. 2019. Abstractive summarization: A survey of the state of the art. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9815–9822. https://doi.org/10.1609/aaai.v33i01.33019815

Lin, Junyang, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. *arXiv preprint arXiv:1805.03989*. https://doi.org/10.18653/v1/P18-2027

Lipton, Zachary C., John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Liu, Yang and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*. https://doi.org/10.18653/v1/D19-1387

Luhn, Hans Peter. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165. https://doi.org/10.1147/rd.22.0159

Luong, Minh Thang, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*. https://doi.org/10.18653/v1/D15-1166

Marsi, Erwin and Emiel Krahmer. 2005. Explorations in sentence fusion. In *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*.

Matousek, Jiri and Bernd Gärtner. 2007. *Understanding and Using Linear Programming*. Springer Science & Business Media.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, Tomas, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2017. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

Miller, George A. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41. https://doi.org/10.1145/219717.219748

Mishra, Aditya Dev and Deepak Garg. 2008. Selection of best sorting algorithm. *International Journal of intelligent information Processing*, 2(2):363–368.

Moawad, Ibrahim F. and Mostafa Aref. 2012. Semantic graph reduction approach for abstractive text summarization. In *Computer Engineering & Systems (ICCES), 2012 Seventh International Conference on*, pages 132–138. https://doi.org/10.1109/ICCES.2012.6408498

Mohan, M. Jishma, C. Sunitha, Amal Ganesh, and A. Jaya. 2016. A study on ontology based abstractive summarization. *Procedia Computer Science*, 87:32–37. https://doi.org/10.1016/j.procs.2016.05.122

Moratanch, N. and S. Chitrakala. 2016. A survey on abstractive text summarization. In *Circuit, Power and Computing Technologies (ICCPCT), 2016 International Conference on*, pages 1–7. https://doi.org/10.1109/ICCPCT.2016.7530193

Nallapati, Ramesh, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence RNNs for text summarization. *CoRR*, abs/1602.06023.

Nallapati, Ramesh, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin. https://doi.org/10.18653/v1/K16-1028

Napoles, Courtney, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100.

Navigli, Roberto. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10. https://doi.org/10.1145/1459352.1459355

Navigli, Roberto. 2012. A quick tour of word sense disambiguation, induction and related approaches. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 115–129. https://doi.org/10.1007/978-3-642-27660-6_10

Nenkova, Ani and Kathleen McKeown. 2012. A survey of text summarization techniques. In *Mining Text Data*. Springer, pages 43–76. https://doi.org/10.1007/978-1-4614-3223-4_3

Over, Paul, Hoa Dang, and Donna Harman. 2007. DUC in context. *Information Processing & Management*, 43(6):1506–1520. https://doi.org/10.1016/j.ipm.2007.01.019

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Pasunuru, Ramakanth and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. *arXiv preprint arXiv:1804.06451*. https://doi.org/10.18653/v1/N18-2102

Paulus, Romain, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. https://doi.org/10.3115/v1/D14-1162

Pilault, Jonathan, Raymond Li, Sandeep Subramanian, and Christopher Pal. 2020. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319. https://doi.org/10.18653/v1/2020.emnlp-main.748

Raganato, Alessandro, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the*

15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 99–110. https://doi.org/10.18653/v1/E17-1010

Rennie, Steven J., Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024. https://doi.org/10.1109/CVPR.2017.131

Rong, Xin. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Rush, Alexander M., Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Sakai, Tetsuya. 2016. Two sample t-tests for IR evaluation: Student or Welch? In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1045–1048. https://doi.org/10.1145/2911451.2914684

See, Abigail, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*. https://doi.org/10.18653/v1/P17-1099

Sharma, Eva, Luyang Huang, Zhe Hu, and Lu Wang. 2019. An entity-driven framework for abstractive summarization. *arXiv preprint arXiv:1909.02059*. https://doi.org/10.18653/v1/D19-1323

Shi, Tian, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. 2018. Neural abstractive text summarization with sequence-to-sequence models. *arXiv preprint arXiv:1812.02303*.

Song, Kaiqiang, Bingqing Wang, Zhe Feng, Liu Ren, and Fei Liu. 2020. Controlling the amount of verbatim copying in abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v34i05.6420

Song, Shengli, Haitao Huang, and Tongxiao Ruan. 2018. Abstractive text summarization using LSTM-CNN based deep learning. *Multimedia Tools and Applications*, pages 1–19. https://doi.org/10.1007/s11042-018-5749-3

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sutskever, I., O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. *Advances in NIPS*.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Watt, Nathan and Mathys C. du Plessis. 2018. Dropout algorithms for recurrent neural networks. In *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 72–78. https://doi.org/10.1145/3278681.3278691

Weischedel, Ralph, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. Linguistic Data Consortium, Philadelphia, PA, 23.

Wolf, Thomas, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

Xu, Song, Haoran Li, Peng Yuan, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Self-attention guided copy mechanism for abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1355–1362. https://doi.org/10.18653/v1/2020.acl-main.125

Yan, Yu, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063*.

Yao, Jin-ge, Xiaojun Wan, and Jianguo Xiao. 2017. Recent advances in document summarization. *Knowledge and Information Systems*, 53(2):297–336. https://doi.org/10.1007/s10115-017-1042-4

You, Yongjian, Weijia Jia, Tianyi Liu, and Wenmian Yang. 2019. Improving abstractive document summarization with salient information modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2132–2141. https://doi.org/10.18653/v1/P19-1205

Yujian, Li and Liu Bo. 2007. A normalized Levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095. https://doi.org/10.1109/TPAMI.2007.1078, PubMed: 17431306

Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zhang, Haoyu, Jianjun Xu, and Ji Wang. 2019. Pretraining-based natural language generation for text summarization. *arXiv preprint arXiv:1902.09243*. https://doi.org/10.18653/v1/K19-1074

Zhang, Jiajun, Yu Zhou, and Chengqing Zong. 2016. Abstractive cross-language summarization via translation model enhanced predicate argument structure fusing. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(10):1842–1853. https://doi.org.1109/TASLP.2016.2586608

Zhang, Yuhui. 2019. Evaluating the factual correctness for abstractive summarization. *CS230 Project*.